Customizable Description and Dynamic Discovery for Web Services

Wooyoung Kim[1], Alan H. Karp
Technical Computing Research Group
HP Laboratories Palo Alto
HPL-2004-45
March 18, 2004*

E-mail: wooyoung@cs.uiuc.edu, alan.karp@hp.com

web services, discovery, ontologies

We present a framework for developing ontologies suitable for a dynamic environment, such as that for web services, and describe its use in a commercial system for resource discovery. This framework recognizes the importance of standards but allows for evolution in a way that doesn't disrupt those adhering to the standards. The framework is based on the notion of discoverable resources which offer extensibility and security. The specific ontology we use in the system includes some salient features, such as attribute based matching rules and the corresponding constraint based search, attributes with dynamic values, and active queries.

* Internal Accession Date Only
[1] Dept of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
To be published in and presented at the ACM Conference on Electronic Commerce (EC'04) 17-20 May 2004, New York, NY

Customizable Description and Dynamic Discovery for Web Services

Wooyoung Kim [*]
Dept. of Computer Science
University of Illinois at Urbana-Champaign
201 N. Goodwin Ave., Urbana, IL, USA
wooyoung@cs.uiuc.edu

Alan H. Karp
Technical Computing Research Group
Hewlett-Packard Laboratories
1501 Page Mill Rd.,Palo Alto,CA,USA
alan.karp@hp.com

ABSTRACT

We present a framework for developing ontologies suitable for a dynamic environment, such as that for web services, and describe its use in a commercial system for resource discovery. This framework recognizes the importance of standards but allows for evolution in a way that doesn't disrupt those adhering to the standards. The framework is based on the notion of discoverable resources which offer extensibility and security. The specific ontology we use in the system includes some salient features, such as attribute based matching rules and the corresponding constraint based search, attributes with dynamic values, and active queries.

1. INTRODUCTION

Since the advent of the telephone company's Yellow Pages, advertisement and discovery have evolved into common components of our day-to-day economic lives. Customers looked up providers by following the particular classification scheme employed in the Yellow Pages. Often, a combination of common sense and a bit of guess work guides the navigation. In the end, users get the contact information for a business or a service. Though any particular lookup may fail and trial-and-error is the number one tactic users adopt, advertisement and discovery using the Yellow Pages have worked remarkably well.

This approach has been successfully carried over to the Web portals, such as Yahoo! Inc., to tackle the advertisement and discovery of Web sites. E-auction sites, such as eBay and QXL, are another example that successfully adopted the Yellow Pages approach. A key characteristic of these sites is that each of them defines its own proprietary classification; providers follow the classification to list their Web site and Web pages, and people find them with an interactive, top-down search strategy using the same classification. It is implicitly assumed that listings are open to public access by anonymous users, and protection of the information of the entries is not a concern to them.

[*]Supported in part by the Defense Advanced Research Projects Agency (Contract numbers: F30602-00-2-0586 and F33615-01-C-1097).

The success of these sites and the pervasive and resilient nature of the Internet have persuaded many infrastructure providers to take a look at the idea of automated business integration on the Internet [10, 7, 5, 23]. As a result, a number of Web standards, such as SOAP, WSDL, WSCL, WSFL, and WS-Inspection, have been developed to make it easy to develop interoperable Web services, and a number of market makers have sprouted to form vertical as well as horizontal marketplaces. It is evident that advertisement and discovery will play a pivotal role in such marketplaces.

Disappointingly, the reaction of market participants to the automated business integration has been lukewarm at the best. The lack of security and interoperability difficulties are blamed for their wariness to the technology. Another factor is the lack of pre-arranged, unambiguous agreement among stakeholders on the meaning of resource and service descriptions. That is because ambiguity or misunderstanding may lead to costly legal entanglement. For example, a price advertised in dollars should not be interpreted in euros by business applications and Web services; common sense or guess work can no longer be tolerated. This agreement on the meaning of terms constitutes an *ontology*.

A number of proposals to create industry-wide or even universal ontologies have emerged during the past five years (*e.g.*, ebXML [10], UDDI [27]). Along with them came public registries where providers advertise their Web services and customers execute queries against it. These efforts are still in development and still lack some critical infrastructural support. One such deficiency is security. These systems control only the insertion and modification of descriptions, but they assume that all data in the repository are visible to anyone. There are many situations, though, where we need to protect not only the services themselves, but also at least some parts of the service descriptions. For example, preferred customers may see prices different from those available to the general public. In general, information should not be leaked between the owner of a resource who defines its attributes and someone who finds the resource. Some of that information may be proprietary. Worse, a malicious user may be able to use the information as the basis of an attack. Conversely, the searcher may not want a malicious provider to see the resource request. For example, a search might include the maximum price the requester will pay for the service.

An ontology system for the Web should have certain characteristics. Resources accessible on the Web are diverse and dynamic; new resources are added, obsolete ones disappear, and existing ones are modified. As a result, resource discovery on the Web must be dynamic. In addition, users should be able to express in a query the exact resources they want to find in a flexible way. It is important that a search request should be able to decide what constitutes a match to protect against an unethical supplier that gets its resource heavy usage by saying it matches every request and from malicious requesters attempting to uncover resources hidden from them. Also, a search request should be able to accommodate the user's preference.

The emergence of numerous vertical markets translates to diverse requirements on the description of resources, which makes it less likely that a single, unified ontology will be able to serve all the different needs in the Internet and on intranets. Indeed, such a global ontology is inherently unscalable, since all parties, even those who never interact, must agree on any change. Instead, we believe there will be marketplaces that develop standard ontologies.

Even in this environment, it is important that those with special requirements should be able to extend these standard ontologies, or even create entirely new ones. This will inevitably lead to fragmentation of Web services descriptions and the issue of interoperability. However, we envision those pieces that get widely used will be incorporated into, or become, standards, while those that aren't can either continue to serve their limited constituency or fade away from lack of use.

We observe that, to be embraced by a wide range of Web participants, an enabling ontology for Web services should facilitate the:

- description of different aspects of a resource/service in sufficient detail and in quantifiable precision. In particular, ontologies should allow resource descriptions to evolve independently of backward compatibility.
- specification of search criteria in such a way as to maximize its discriminating power and minimize false hits.
- specification and enforcement of visibility control measures. Users should not get even a hint of the existence of services that they are not authorized to see.
- expression of domain-specific knowledge. The ontology specification needs to be independent of syntax/representation so that it can be implemented using the best technology available.
- seamless integration of different registries.

Creation of such ontologies will encourage market participants to advertise their Web services, fostering automated service integration on the Internet and creating new business opportunities.

In this paper we describe a service description and discovery framework (SDDF) which addresses the requirements described above. We developed the framework in the context of Hewlett-Packard's e-speak platform, one of the first environments for Web services. Being one of the core services, the service description and discovery has been discussed in several papers and articles related to e-speak (e.g., see [13, 14, 17]). However, the discussion has been limited in scope as most of them framed it as a component service of a larger e-services infrastructure. We believe this paper is the first attempt to present a holistic discussion of e-speak service description and discovery. For a more general discussion of e-speak in the e-commerce and e-services perspective, including the origin, the vision, the requirements, and its other core services, we refer readers to [16].

The key aspect of the e-speak approach is the notion of a *vocabulary as a discoverable resource*, where a vocabulary represents an ontology as a set of attribute-value pairs. The framework consists of both resource description and discovery models [1]. The approach described in this paper proved highly effective in resource advertisement and discovery.

2. ONTOLOGY AS A DISCOVERABLE RESOURCE

As used here, an ontology is an abstract agreement among users to create and advertise resource descriptions and to discover resources. It is a succinct, unequivocal specification of how to interpret resource descriptions and search queries specified in it.

A way to simplify managing ontologies is to make them discoverable resources. Surprisingly, conferring on ontologies these two properties, *i.e.*, *being discoverable* and *being a resource*, turns out to give ontologies a number of desirable properties. First, anyone who can create a resource can create an ontology, democratizing the process of ontology creation. Now, users need not wait until an external entity (most likely, a standardization organization) makes appropriate changes to an ontology to accommodate their special requirements. Consider, for example, a provider of a specialized service with features not included in any standard ontology. All that's needed is to create a new ontology capable of describing these features and to advertise that ontology as a discoverable resource in the standard ontology. Someone doing a search will find resources and, perhaps, this ontology. Anyone knowing how to use the ontology can then extend the search to find the advertiser's specialized service. Of course, this approach can lead to fragmentation of the marketplace; there is a good chance that an individual won't know how to use a specialized ontology discovered in a search. Standardization is crucial for interoperability. Hence, we still envision that standards bodies

[1]A *resource* denotes a uniform internal representation of an entity created in or registered with the e-speak system. In the rest of the paper we use *resources* not just to represent Web services but rather to represent inclusively all things accessible on the Web, ranging from Web services to e-services to data to storage to CPU cycles.

such as the WWW Consortium [31], RosettaNet [23], CommerceNet [5], UDDI [27], and ebXML [10], will create a few globally adopted market-specific ontologies. However, it is also our belief that standardization should not necessarily stifle diversity among individuals' requirements.

Being discoverable introduces an interesting recursion for ontologies. That is, in order to be discovered an ontology must be described in some other ontology. To ground the recursion, a special ontology is required which is uniquely identified across platforms without explicit discovery. For example, the e-speak system defined the `Base` vocabulary which was unique across all e-speak engines. Note that the knowledge of the "base" vocabulary must be communicated among participants before any meaningful interaction can occur. Fortunately, this vocabulary can be extremely simple since it should be easy to create more advanced vocabularies. For example, the UDDI geographic taxonomy could be described in a base vocabulary as having the attribute-value pairs `Name=UDDI` and `Type=Geo`.

An interesting observation is the analogy of ontologies as resources to types in programming languages. As a type defines a set of values in the type, an ontology defines a set of descriptions in it; an ontology naturally demarcates resources of a particular kind. Hence, resource descriptions in a set of ontologies provide a natural type system, in the same way that classes partition objects by type.

Treating ontologies as discoverable resource allows security policies, such as visibility and access control, to be enforced on them. Since processes acting on behalf of users are granted specific rights, those that are denied access to certain ontologies have no way to find resources advertised in those ontologies. Furthermore, hierarchically organizing ontologies gives rise to hierarchical control of resource visibility, greatly simplifying the management of security policies.

2.1 Ontology Creation

Ontologies themselves should not depend on what resources should or should not be. This independence allows ontologies to take any representation as long as it is suitable for the domain of interest.

The e-speak system used vocabularies, sets of attributes, to represent ontologies. Name-value pair attributes are one of most general forms of metadata representation. Resource descriptions are defined as a set of name-value pairs and are selected with predicates made of attribute names and values. The e-speak system implemented two vocabulary specifications, one in Java and the other in XML. All examples in this paper show the XML version.

A vocabulary in e-speak defines attributes that are used for resource description. Each attribute has the following properties:

Name: A string different from the name of any other attribute in this vocabulary.

Type: The value type of this attribute, *e.g.*, integer, string.

Multiplicity: Either *single* or *multiple*.

Value: Zero or more values of the specified type.

Matching rule: A function defining what constitutes a match.

Mandatory: True if at least one value must be supplied for this attribute.

Must-match: True if a look-up will succeed only if this attribute matches.

The vocabulary developer decides on these properties. Anyone using the vocabulary, either to advertise a service or to discover one, must follow the rules of the vocabulary being used.

The matching rules are specified per attribute rather than per value-type. The latter approach is too limiting. For example, the waist and inseam of a pair of pants may both be represented by integers, but the vocabulary developer may know that the waist attribute requires an exact match while inseams that differ by one unit may still fit. Hence, attribute-based matching rules allow more flexibility.

The matching rules are specified in terms of components defined in the vocabulary definition language, *e.g.*, equality testing on standard data types, lexical order rules for strings, sorting rules for integers, *etc.* In order to prevent denial of service attacks against the party doing the matching, the language for specifying these rules must be loop-free [2]. Figure 1 shows the XML Schema definition for vocabulary creation in e-speak.

2.2 Dynamic Attributes

Attributes in a resource description are typically independent of each other. They are bound to values and remain unchanged until they are explicitly modified. As opposed to these early binding attributes, some attributes are indeed dependent on other attributes and can be specified in a well-defined closed-form expressions (*i.e.*, *late binding*). Some of them may even depend on dynamically changing attributes, such as network load or stock price. Such attributes need to be computed at access time. We call them *dynamic attributes*.

Resource providers may use dynamic attributes to give customers a different presentation depending on their profile information. For example, the price of a product specified in dollars can be delivered to Eurozone customers in euros. A company may want customers to get a different discount price based on the current discount rate. The discount price is dependent on the discount rate at the time of the lookup/access. For example,

```
<attribute name="discount_price" type="dynamic"
           default="price">
  <variable name="x" src="..."/>
  <expression> price * $x/rate </expression>
</attribute>
```

Note that the rate may change at any time. Also, it usually changes more frequently than the retail price of the product. By specifying an expression that computes the discount

[2] "If your program has no loops, then it's done." – Donald Knuth.

```
<?xml version="1.0"?>
<schema targetNameSpace=
          "http://www.e-speak.net/e-speak.vocab.xsd"
        xmlns="http://www.w3.org/1999/XMLSchema"
        xmlns:es-vocab=
          "http:/www.e-speak.net/e-speak.vocab.xsd">
 <element name=attrGroup minOccurs=1>
  <complexType>
   <element name=attribute minOccurs=1
          maxOccurs=unbounded content=elementOnly>
    <complexType>
     <element ref=es-vocab:datatypeRef>
     <attribute name=required type=boolean
               use=default value=false />
     <attribute name=multivalued type=boolean
               use=default value=false />
     <attribute name=name type=string use=required />
    </complexType>
   </element>
   <attribute name=name type=string />
  </complexType>
 </element>
 <element name=datatypeRef>
  <complexType>
   <element name=default type=string minOccurs=0
          maxOccurs=1/>
   <element name=minInclusive type=string minOccurs=0
          maxOccurs=1/>
   <element name=maxInclusive type=string minOccurs=0
          maxOccurs=1/>
   <attribute name=name  type=string use=required/>
   <attribute name=mustmatch type=boolean use=default
          value=false />
  </complexType>
 </element>
 <complexType name=vocabularyDecl content=empty>
  <attribute name=src type=uri-reference use=required/>
 </complexType>
</schema>
```

Figure 1: XML schema for vocabulary creation

price from its retail price, the company saves updating discount prices of their products as the discount rate changes. The update to the discount rate suffices. Another benefit of using dynamic attributes is that it is more economical to enforce access control on discount rates than discount prices, as the number of resources are much larger than that of discount rates.

2.3 Vocabulary Translators

An effective way to alleviate the interoperability problem in vocabularies is vocabulary translation services. A vocabulary translator maps a description in the source vocabulary to one in the target vocabulary. For example, it may translate the words in an attribute from French to German or from Unix to MVS. Vocabulary translation is not necessarily one-to-one. A translator may ignore certain attributes from the source vocabulary, or may fill in some attributes in the target vocabulary with default values. The vocabulary translation service can even be out-sourced and provided by a third-party.

2.4 Example: Tennis Shoe Vocabulary

We illustrate how new vocabularies might be seamlessly integrated into existing ones and how vocabulary translators can facilitate the process.

Say that a shoe company has a new kind of tennis shoes to advertise that doesn't fit any existing vocabulary. There is a "tennis shoe" vocabulary that is already being widely used. All it needs to do is create its "clay court shoe" vocabulary and advertise it in the "tennis shoe" vocabulary. People searching that vocabulary will find tennis shoes and the new vocabulary. Because the schema is open for examination, anyone interested in playing on clay courts can then search using the "clay court shoe" vocabulary.

Of course, different countries have different units for measuring shoe size. The company could create a separate vocabulary for each case, but it might not even know all the possibilities. Instead, it may create the "clay court shoe" vocabulary using a specific unit and advertise it with a reference to a translation service which is offered by a third party, perhaps someone who extended the base shoe vocabulary. People searching with the "clay court shoe" vocabulary who use different units can feed the translation service with their look-up requests before using it for a search. Note that query translation need not necessarily be exposed to end users.

3. RESOURCE DESCRIPTION WITH VOCABULARIES

The metadata associated with a resource is split into two parts, the resource specification and the resource description. The resource specification contains information on access, management, and protection of the resource; it constitutes the private part of resource's metadata. The resource description lists different aspects of the resource; it is the publicly searchable part. Users specify queries against the resource description to discover the resource, while actual access is regulated according to its resource specification. This dichotomy of resource metadata allows the access rules and the vocabularies to evolve independently.

The following XML document is a registration of a resource known as `Box-A` and as `Container-C`. Note that two vocabularies `box` and `container` are used to specify its resource description.

```
<?xml version="1.0"?>
<resource xmlns=
  "http://www.e-speak.net/Schema/Espeak.register.xsd">
 <resourceSpec>
  <interface>...</interface>
  <security>...</security>
  <filter>...</filter>
 </resourceSpec>
 <element xmlns:box="http://www.parcel.com/box.xsd">
  <box:Name> Box-A </box:Name>
  <box:Length> 20.0 </box:Length>
  <box:Width> 25.5 </box:Width>
  <box:Height> 60.0 </box:Height>
 </element>
 <element xmlns="http://www.parcel.com/container.xsd">
  <Name> Container-C </Name>
  <Volume> 30600.0 </Volume>
  <Content> bubble gum </Content>
 </element>
</resource>
```

Four components, namely interfaces (`interface`), security policy (`security`), filter constraints (`filter`), and resource specific data (`rsd`), comprise the resource specification of a

```
<!ELEMENT resource (resourceSpec, element*)>
<!ELEMENT resourceSpec (interface?, security?,
                        filter?, rsd?)>
<!ELEMENT filter (condition)>
<!ELEMENT rsd (private?, public?)>
<!ELEMENT private (#PCDATA)>
<!ELEMENT public (#PCDATA)>
```

Figure 2: XML Specification for Resource Description. A filter is given as a condition in the e-speak constraint language (`condition`).

resource (Figure 2). The `interface` element specifies how clients access the resource. It may be an instance of ESIDL (the e-speak interface definition language), a Java interface, or a WSDL document. How to enforce access and visibility control for the resource is laid out in the `security` element. For example, it prescribes under what conditions access to the resource is granted.

Resource providers use the `filter` element to specify matching rules to screen users whom they don't want to discover the resource. The `rsd` element may have other data pertinent to the resource. The data may be public and visible to general users or it may be private data which only the owner can access. Although interface and security are critically important for Web services provisioning and management, further discussion on them is beyond the scope of the paper. Interested readers are advised to consult other published e-speak documents, such as SFS [1] and the e-speak architecture specification [15] for the details.

Many resources assume different roles depending on their functionality or environment. Consequently, they may have multiple descriptions. For example, a fax machine can serve as a facsimile transmitter as well as a telephone. Hewlett-Packard Company manufactures PCs and servers for computer resellers as well as individual customers. To build such systems, they purchase microprocessors from Intel Inc., for example. To accommodate the requirement of expressing multiple aspects of a resource, the e-speak SDDF allows the resource description of a resource's metadata to have multiple descriptions.

The resource description consists of a set of description elements in different vocabularies, each of which is a set of name-value pairs and describes a specific aspect of a resource. To facilitate description validation, a description element is required to refer to the vocabulary in which the attributes are defined.

```
<resource xmlns=
  "http://www.e-speak.net/Schema/Espeak.register.xsd">
 <resourceSpec> ... </resourceSpec>
 <element vocabulary="personal computers">
  <manufacturer>Hewlett-Packard</manufactuere>
  <type>notebook PC</type>
  <model-name>Compaq Presario</model-name>
  <model-no>2555us</model-no>
  <price>999.99</price>
 </element>
 <element vocabulary="Rebate">
  <amount>100.0</amount>
  <kind>mail-in</kind>
  <good-until>
   <date>3</date><month>May</month><year>2003</year>
  </good-until>
 </element>
</resource>
```

For example, the above registration describes a Hewlett-Packard Compaq Presario notebook PC whose price is $999.99. It also describes that the resource comes with a $100.00 mail-in rebate offer that ends May 3, 2003. A vocabulary is referenced by a name, which may be a URN. The registration process will associate the specified designation with a resource representing a vocabulary. The metadata associated with that vocabulary resource allows verification of the requester's access rights and validation of the description elements against the corresponding vocabulary before registering the resource.

3.1 Matching Rules

A matching rule determines when the component of a query matches some aspect of a resource description. It uses such things as integer relational operations and collating sequences. Common matching rules are integer equality and substring equivalence, but they can be more complex to allow for user defined types. For example, a matching rule on a `Bigint`, which consists of two integers, might report a match if both integers in the query equal the corresponding integers in the description.

The availability of vocabulary translators makes defining matching rules easier. Consider the tennis shoe vocabulary example in Section 2.4. In US units, a shoe $\pm\frac{1}{2}$ the requested size might fit, but in British units the leeway is ± 1. Since each attribute has an associated matching rule, knowing the units of the request makes it easier to write the matching conditions, while having translators allows that rule to correctly match requests originally expressed in a variety of units.

3.2 Filters

Filters are to resource providers what queries are to requesters. They are matching rules that resource providers use to specify those queries they want to allow to match their resource descriptions based on the rights of the requester. Therefore, a filter is specified at the individual resource level, as opposed to a matching rule specified at the vocabulary level.

A filter is a predicate over attributes from the description of a resource and profile information of the user. It is assumed that each requester is represented as a resource associated with a resource description in one or more vocabularies. Only those resources with filter constraints that evaluate to true are included in the result set.

Some of requirements that can be expressed with filters are:

- A mortgage broker may specify that only clients with good credit history find mortgage programs with the preferred interest rate.

- A chip design company may specify that only specialty chip designers find very high-resolution plotters and printers.

```
<resourceSpec>
 <filter><condition test=
     "\$user.loc/loc_in_building='west'"/></filter>
</resourceSpec>
<element xmlns:printer=
            "http://www.hp.com/e-speak/Printer.xsd">
 <printer:manufacturer> HP </printer:manufacturer>
 <printer:model> LaserJet III </printer:model>
 <printer:name> nile </printer:name>
</element>
```

Figure 3: A filter constraint specifying that printer 'nile' be visible only to the employees in the west wing of the building. A meta variable $user refers to the profile document of a requester.

- An organization may specify workers in the west wing may find only printer "nile" while those in the east wing find only printer "ganges" (Figure 3).
- A service provider may specify that only customers within the US will find a particular service.

3.3 Extending a Vocabulary

The e-speak vocabulary framework provides no direct support for extending a vocabulary. Instead, users can simulate vocabulary extension using existing abstractions of vocabulary creation and resource description; creating a new vocabulary with additional attributes and letting it have a reference to the vocabulary that it extends effectively amounts to extending a vocabulary. New resources described in the extending vocabulary will be given descriptions in the extended vocabulary as well as in the newly created vocabulary. For ease of use, an additional service can be arranged, which presents a virtual vocabulary composed of an extending and an extended vocabulary.

Note the similarity of the extension by creation to delegation in object-oriented programming languages. As in object-oriented programming languages, inheritance is an obvious alternative. Indeed, RDF supports a fine-grained schema extension using multiple inheritance [38]. Answers to a question of which one is better in the context of ontology extension are yet to be seen. Nonetheless, we note several advantages of the extension by creation over inheritance. First, the extension by creation is less obtrusive – little change needs to be made to the vocabulary infrastructure as well as resource registration/discovery (in particular, existing resource descriptions). As a result implementation is very easy. Many standard ontologies including UDDI can be easily extended and customized without any specialized support. Also, backward compatibility comes for free: resources that were found before are still found with the same vocabulary after the extension.

4. RESOURCE DISCOVERY WITH VOCABULARIES

Web services may enter and leave e-marketplaces dynamically. New resources become available, obsolete ones are no longer accessible, old ones are gradually retired, and existing ones modified. Consequently, the same lookup request may return different results at different times. The practice of discovering resources before accessing them helps ensure that resources found are up-to-date and available. One implicit assumption in the lookup process is requesters are no longer assumed to be either anonymous or benevolent. Their identities and credentials are thoroughly verified, their access rights rigorously enforced, and their profile information securely maintained and managed by the underlying system.

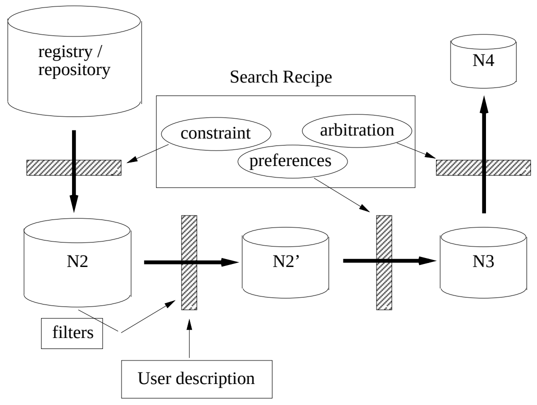

Figure 4: E-speak Lookup Process. The constraint is evaluated against the registry/repository to generate $N2$. At this moment if a resource in $N2$ has an associated filter, it is applied against the requester's user descriptions, further sifting the results. If none has a filter, $N2 = N2'$. If preferences are given they are applied to $N2'$ to order the results. If the requester specified an arbitration statement, it is applied to $N3$ to further screen the results. With no arbitration policy, $N3 = N4$. Note $|N2| \geq |N2'| = |N3| \geq |N4|$.

Users discover resources by looking up in a repository (or possibly, repositories) with a query. How repositories are organized is not important to queries as long as the underlying system understands the query and returns matching resources to the requester. A query is constructed with attribute names from different vocabularies and desired matching values. (Recall that a resource may have multiple description elements.) Having vocabulary references in a query has a number of benefits. For example, it facilitates query validation. Also, by qualifying attribute names with a vocabulary identifier we can avoid potential name conflicts among attributes from different vocabularies. Requiring resource lookup be subject to vocabularies confers more discriminating power to queries. A user trying to find a 21-inch TV will never get a 21-inch monitor. More importantly, it allows economical visibility control on resources. Having no access rights on a vocabulary means one cannot find any resources described in the vocabulary.

In the e-speak SDDF, lookup queries are implemented with a search recipe. A search recipe is composed of vocabulary declarations, a constraint, preferences, and an arbitration statement (Figure 4). The constraint specifies a condition that resources of interest must satisfy. Thus, a constraint must be present in a search recipe. The preferences are collectively applied to put the results in the preferred order while the arbitration statement determines which results are returned to requester. The last two are optional.

4.1 Vocabulary Declaration

A vocabulary declaration associates a local vocabulary identifier with a vocabulary reference. A local vocabulary identifier is prefixed to an attribute name to refer to the attribute unequivocally. A vocabulary declaration is specified with the `vocabulary` element which defines `name` and `src` attributes. The `name` attribute specifies a local vocabulary identifier while `src` specifies how to get the vocabulary. The value of `src` may be the name with which the vocabulary is registered, another query, or a URI which uniquely points to the schema definition of the vocabulary (e.g., XML DTD or XML Schema). Omitting the name attribute declares a default vocabulary; attributes with no qualifying prefix are assumed to be from the default vocabulary. Only one default vocabulary is allowed in a query. For example,

```
<vocabulary name="seller" src="PC seller" />
<vocabulary name="buyer"
            src="http://www.CPU.org/CPUbuyer.xsd"/>
<vocabulary src="http://www.parcel.com/box.xsd"/>
```

4.2 Constraint

A constraint is a predicate over attributes that may be defined in different vocabularies. For example, the following query returns a list of PC manufacturers who sell their products to `ResellerA` and buy CPUs from `CPUMakerB`.

```
<?xml version="1.0"?>
<esquery xmlns=
   "http://www.e-speak.net/Schema/Espeak.query.xsd">
 <vocabulary name="seller" src="PC seller" />
 <vocabulary name="buyer" src="CPU buyer"/>
 <constraint>
  seller:customer/name='ResellerA' and
              buyer:supplier/name='CPUMakerB'
 </constraint>
</esquery>
```

Identifiers in the constraint, such as `customer` and `supplier`, refer to attributes in the vocabularies. An attribute may have an arbitrary XML document as its value. XPath [32] expressions are used to refer to individual parts in the document. The name resolution with XPath-based constraints is sufficient for our purpose. (For more formal and comprehensive reasoning on name resolution, refer to [4]).

4.3 Preference

Conditions that matching resources should satisfy are expressed in the constraint. By contrast, preferences represent additional properties that users want the matching resources to have. More than one preference is allowed. The preferences in a search recipe are collectively evaluated and the matching resources are ordered according to the total preference score.

`Min` and `max` are the two basic preference operators. The former specifies ordering of resources in ascending order of the evaluation result of the expression while the latter does the same in descending order (Figure 5 (a)). If multiple `max`/`min` preferences are specified in a search recipe, the order that the preferences appear becomes significant. The first one provides the primary order, the second one is used as a tie breaker for the result of the first, and so on.

More involved preferences are expressed using the `with` operator. This operator takes a condition and a weight. The condition is a predicate over attributes which may come from different vocabularies. A user may specify multiple `with` preferences (Figure 5 (b)).

```
<vocabulary src="http://www.parcel.com/box.xsd"/>
<preference operator="max" >
 <expression>Length*Width*Height</expression>
</preference>
```

(a) A `max` preference.

```
<?xml version="1.0"?>
<esquery xmlns=
  "http://www.e-speak.net/Schema/Espeak.query.xsd">
 <vocabulary src="box" />
 <constraint> color = 'blue' </constraint>
 <preference operator="with" >
  <condition> Length &lt; 5 </condition>
  <weight> 2 </weight>
 </preference>
 <preference operator="with">
  <condition> Height &gt; 7 </condition>
  <weight> 4*Height </weight>
 </preference>
</esquery>
```

(b) `with` preferences.

Figure 5: Preference examples. (a) The preference says the user prefers bigger boxes. (b) A user prefers boxes whose length is less than 5 and whose height is greater than 7, but she prefers taller ones.

For a matching resource, if the condition of a `with` preference evaluates to true, the accompanying weight is added to the total score of the resource. If evaluated to false nothing is added. After all the `with` preferences are evaluated the resources are ordered in decreasing order of their total weight.

In cases where the `with` and `min`/`max` preferences appear together in a search recipe, the `with` preferences are given a higher priority. Thus, the lookup results are ordered according to the `with` preferences, first. Then, the first `min` or `max` preference is used to break ties, the second is used to break ties from applying the first, and so on.

4.4 Arbitration

Arbitration specifies which resources among the matching resources are to be returned to the requester. It is similar to the return cardinality policy in Object Management Group Trading Object Services Specification [20]. The operators supported are:

- `first` takes an integer argument n, and returns up to n resources.
- `all` takes no argument and returns all matching resources.
- `any` takes no argument and returns any matching resource.

```
<arbitration operator="first" cardinality="3" />
<arbitration operator="all" />
<arbitration operator="any" />
```

The resource returned with any is implementation-dependent. A repository can reduce its workload by returning the first resource matching the request. On the other hand, returning a random matching request is one way to statistically load balance requests among a set of identical resources.

4.5 Active Queries

A typical lookup process (database access, Web services lookup, etc) follows a request-response model; when a user sends a lookup request, the system returns matching resources, if any. If none is found, the system may return an empty set and discard the request. By contrast, *active* queries remain in the system until a match is found. Active queries are associated with an expiration time (or time-to-live) and the system keeps them until they expire. If matching resources are found before the expiration, they are sent back to the requester. Depending on user's requirement, the system may remove the lookup request once any match is found, or may keep it until its expiration time.

Active queries are quite useful when requesters need to poll repositories periodically to find a resource or Web service. For example, suppose Larry wants to purchase a left handed Swiss Army Knife from an e-auction site. Without active queries he has to periodically visit the site and look through the listing to find out if the knife is listed. By contrast, an active query waits until the item is posted and then notifies Larry of of the availability of the item. E-speak provided a similar function to active queries as part of its event service.

5. RELATED WORK

Web services have been touted for quite a while as the next "Big Bang" in the market. As such, we witnessed no shortage of research activities revolving around ontology development for Web services. We review a few preeminent ontologies and ontology frameworks for Web services that are currently being actively developed in industry and academia.

ebXML [10] is an international initiative established by UN/CEFACT [29] and OASIS [21] to standardize XML business specifications. The ebXML architecture [11] defines registration and discovery mechanisms [19] based on an ontology [18] along with information registry/repository which stores ebXML Core Components, Collaboration-Profile Protocols (CPP's) and Collaboration-Profile Agreements (CPA's). A handful of open source ebXML projects (*e.g.*, [26] and [25]) have been launched to build reference implementations of ebXML component services.

UDDI [27] is an industry initiative to build a public Web services clearing house. It was initiated by Ariba, IBM, and Microsoft, and later joined by Hewlett-Packard; it is now operating under the auspice of OASIS. A few public registries called operator sites are maintained for companies to publish and discover Web services. A core component of the UDDI technology [28] is registration which contains "white pages" for name, address, contact information, and other identifiers; "yellow pages" for classification of a business under standard taxonomies; and "green pages" that contain technical information about the Web services being described. UDDI also lists a set of APIs for publication and inquiry. Unlike e-speak vocabularies, the UDDI publication APIs define a set of fixed attributes for describing Web services. Similarly, the UDDI inquiry APIs accept only rigid queries (e.g., some attributes are associated with implicitly specific operators) which have references to the attributes defined in the UDDI publication APIs and do not support "rich" queries, limiting flexibility and extensibility. The syntax of the APIs is specified in UDDI API Schema [22].

The Java API for XML Registries (JAXR) is designed to provide a uniform API for accessing information in different registries from within the Java platform. Currently it supports ebXML and UDDI registries. JAXR consists of a set of Java interfaces through which interested parties share information. Though it provides interoperability between disparate registries to some extent by hiding the differences from users, it takes an inherently ad hoc and language-dependent approach and thus may have only limited success. Emergence of a new registry will induce substantial modification or extension to the JAXR implementation [24, 25].

The Semantic Web [3] is attempting to extend the current Web by creating machine-understandable Web content consisting of properties and relationships of Web resources including Web services. In the Semantic Web, information is given well-defined meaning, enabling more effective discovery, automation, integration, and reuse across Web resources. No separate registries are assumed; the Web itself serves as the registry. The Semantic Web is being built upon three key Web standards, namely, the Universal Resource Identifier [2], the eXtensible Markup Language [36], and the Resource Description Framework (RDF) [38, 37]. In particular, the RDF Schema [37] and the Web Ontology Language (OWL) [30] are being developed that help creating Web ontologies which define classes of resources and relations among them as well as a set of inference rules. In addition, query languages, such as DAML Query Language [8], are being developed to facilitate automated knowledge discovery in the Semantic Web. Ontologies themselves are managed as resources. Interoperability between different but similar ontologies is supported by encoding equivalence information within the description of an ontology.

A closely related line of research is to develop an OWL-based Web Service Ontology, OWL-S [9], (formerly DAML-S: DAML-based Web Service Ontology) which defines a specific ontology "for describing the properties and capabilities of Web services in unambiguous, computer interpretable form." In OWL-S, a service is described by an instance of the class SERVICE – each instance "*presents* a descendant class of SERVICEPROFILE, is *describedBy* a descendant class of SERVICEMODEL, and *supports* a descendant class of SERVICEGROUNDING." This SERVICE class corresponds to the resource in e-speak. The service profile contains information that an agent uses to discover the service; it provides largely what the *resource description* in e-speak provides for a service. The service model describes advertised functions of a service by specifying its inputs, outputs, preconditions and effects; thus, it plays a similar role to what the resource specification in e-speak serves. A service grounding specifies detailed information that an agent uses to access a service, such as communication protocols, message formats, and port numbers. Web Services Description Language (WSLD) [34] is chosen as the initial implementation language for service groundings. In contrast, e-speak allows such information to be specified independently of SDDF, for example, using a separate specification language, such as Web Services Conversation Language (WSCL) [33] and WSDL.

The Open Grid Services Architecture (OGSA) [12] is an attempt to align Grid and Web services technologies. In OGSA, a Grid is defined as an extensible set of Grid services which are in turn defined as a Web service that provides a set of well defined interfaces and that follows specific conventions. All Grid services are required to implement the GridService interface which defines the FindServiceData operation, among others. Grid services are described using WSDL [34] and registered in a registry service with their *service data*, a set of named and typed XML elements (service data elements) representing service metadata. Any Grid service can be a registry service as long as it implements the Registry interface. Service discovery is done through the FindServiceData operation with a simple "by name" query language. The operation can be extended to allow for more involved query language such as XQuery [35].

6. DISCUSSION

Vocabularies as described here have some interesting properties.

- Supporting mandatory attributes adds an operational aspect to the ontology that goes beyond the semantic aspect we commonly think of.
- Must-match attributes provide password access to the discovery of resources.
- The ability to specify matching rules, particularly attribute-based rules, adds a different kind of semantic content to the vocabulary.
- Support for constraint-based matching allows more control to searchers.
- Arbitration policies greatly simplify implementing the client side.

The most interesting property was unintentional. Vocabularies are discoverable resources because everything in e-speak was a discoverable resource. It was only after we built the system did we realize that this property gave us dynamically extensible ontologies. E-speak vocabularies did not support inheritance; rather, the extensibility came from delegation which occurred quite naturally by advertising some attributes in the "parent" vocabulary and others in the "child" vocabulary. Another shortcoming was the problem of *vocabulary unification.* Two parties could produce identical vocabularies and advertise them both in a standard vocabulary. Someone searching for a service advertised in these vocabularies had to supply two, identical search expressions, one for each vocabulary. The system should have been able to unify these vocabularies, but we never were able to solve the evolution problem. If the system merged the two vocabularies, they could only change if both parties coordinated their activities. We felt that redundant search expressions and null translation services were less problematic.

Nonetheless, e-speak vocabularies proved to be sufficiently flexible to allow the evolution of standard ontologies in a dynamic environment. Small communities could extend the standards without needing approval from the larger set of users or the respective governing bodies. We believe that, had e-speak continued its existence long enough, we would have seen an ecosystem of vocabularies. Those that proved useful would become standards, or would have their extensions merged into the standard vocabularies. Those that didn't have sufficient value would disappear or continue to serve their small communities.

We are confident that what we've presented works and is useful. Everything described here, except a few features that we didn't get a chance to implement, such as active query and attribute-based matching rules, has been implemented in e-speak and used by our customers. We also made available a UDDI-compatible private registry called e-Services Village [6], which used vocabularies to add dynamically extensible, rich query to the fixed taxonomies of UDDI.

7. REFERENCES

[1] A. Banerji, C. Bartolini, D. Beringer, A. J. Boulmakoul, S. Frølund, K. Govindarajan, A. Karp, M. Morciniiec, G. Pogossiants, C. Preist, S. Sharma, D. Stephenson, and S. Williams. Service Framework Specification, Part 1. Version 2.0. Technical Report HPL-2001-138, Hewlett-Packard Laboratories, 2001.

[2] T. Berners-Lee, R. Fielding, and L. Masinter. *Uniform Resource Identifiers (URI): General Syntax*, 1998. Network Working Group RFC 2396. http://www.ietf.org/rfc/rfc2396.txt?number=2396.

[3] T. Berners-Lee, J. Hendler, and O. Lassilal. The Semantic Web. *Scientific American*, 284(5), May 2001. http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2.

[4] M. Bowman, S.K. Debray, and L.L. Peterson. Reasoning About Naming Systems. *Transactions of Programming Languages and Systems*, 15(5):795–825, 1993.

[5] CommerceNet. http://www.commerce.net/.

[6] Hewlett-Packard Company. E-Services Village, 2002. ftp://ftp.hp.com/pub/linux/espeak.

[7] Microsoft Corp. BizTalk^{TM} Framework 2.0: Document and Message Specification, December 2000. http://www.microsoft.com/biztalk/.

[8] DAML Query Language, 2003. http://www.daml.org/dql/.

[9] DAML Services. http://www.daml.org/services/.

[10] ebXML. http://www.ebxml.org/.

[11] ebXML. ebXML Technical Architecture Specification v1.0.4, 2001. http://www.ebxml.org/specs/ebTA.pdf.

[12] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. *The Physiology of the Grid – An Open Grid Services Architecture for Distributed System Integration*, June 2002. http://www.globus.org/research/papers.html.

[13] S. Graupner, W. Kim, D. Lenkov, and A. Sahai. E-speak: an Enabling Infrastructure for Web-based E-services. In *Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR '00)*, l'Aquilla, Italy, August 2000. http://www.ssgrr.it/en/ssgrr2000/papers/134.pdf.

[14] S. Graupner, W. Kim, A. Sahai, and D. Lenkov. E-speak - an XML based Document Exchange Engine. In *Proceedings of the 2nd International Conference on Electronic Commerce and Web Technologies (EC-Web '01)*, pages 270–279, 2001.

[15] Hewlett-Packard. e-Speak Architectural Specification, June 2001. Release A.03.13.00.

[16] A. Karp. E-speak E-xplained. *Communications of ACM*, 46(7):112–118, July 2003.

[17] W. Kim, S. Graupner, A. Sahai, D. Lenkov, C. Chudasama, S. Whedbee, Y. Luo, B. Desai, H. Mullings, and P. Wong. Web E-Speak: Facilitating Web-Based E-Services. *IEEE Multimedia*, 9(1):43–55, January-March 2002. http://csdl.computer.org/comp/mags/mu/2002/01/u1043abs.htm.

[18] OASIS/ebXML Registry Technical Committee. OASIS/ebXML Registry Information Model v2.5, June 2003. http://www.oasis-open.org/committees/regrep/documents/2.5/specs/ebrim-2.5.pdf.

[19] OASIS/ebXML Registry Technical Committee. OASIS/ebXML Registry Services Specification v2.5, June 2003. http://www.oasis-open.org/committees/regrep/documents/2.5/specs/ebrs-2.5.pdf.

[20] Object Management Group. Trading Object Services Specification. Version 1.0. http://www.omg.org/docs/formal/00-06-27.pdf.

[21] Organization for the Advancement of Structured Information Standards. http://www.oasis-open.org/.

[22] Organization for the Advancement of Structured Information Standards. UDDI API Schema, 2003. http://uddi.org/schema/uddi_v3.xsd.

[23] RosettaNet. http://www.rosettanet.org.

[24] Sun Microsystems, Inc. Java Web Services Developer Pack v1.3, 2004. http://java.sun.com/webservices/webservicespack.html.

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

[26] The Open ebXML Project. http://openebxml.sourceforge.net/.

[27] UDDI. http://www.uddi.org/.

[28] UDDI Spec Technical Committee. *UDDI Version 3.0.1: UDDI Spec Technical Committee Specification*, 2003. http://uddi.org/pubs/uddi-v3.0.1-20031014.pdf.

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

[30] Web Ontology Working Group. OWL Web Ontology Language. http://www.w3.org/2001/sw/WebOnt/.

[31] World Wide Web Consortium. http://www.w3c.org/.

[32] World Wide Web Consortium. *XML Path Language (XPath) Version 1.0*, 1999. W3C Recommendation 16 November 1999. http://www.w3.org/TR/xpath.

[33] World Wide Web Consortium. *Web Services Conversation Language (WSCL) 1.0*, 2002. W3C Note. http://www.w3.org/TR/2002/NOTE-wscl10-20020314/.

[34] World Wide Web Consortium. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, 2003. W3C Working Draft. http://www.w3c.org/TR/wsdl20/.

[35] World Wide Web Consortium. *XQuery 1.0: An XML Query Language*, 2003. W3C Working Draft. http://www.w3.org/TR/xquery/.

[36] World Wide Web Consortium. *Extensible Markup Language (XML) 1.0 (Third Edition)*, 2004. W3C Recommendation. http://www.w3.org/TR/2004/REC-xml-20040204/.

[37] World Wide Web Consortium. RDF Vocabulary Description Language 1.0: RDF Schema, 2004. W3C Recommendation. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[38] World Wide Web Consortium. Resource Description Framework (RDF): Concepts and Abstract Syntax, 2004. W3C Recommendation. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.