UUDIe: An Extended Registry for Web Services

Omer F. Rana
School of Computer Science
and Welsh eScience Centre
Cardiff University
o.f.rana@cs.cf.ac.uk

http://www.cs.cf.ac.uk/User/O.F.Rana/
http://www.wesc.ac.uk/
Overview

• UDDIe
  - extensions to UDDI to help us do what we want
  - support for metadata annotations

• Based on discussions with Luc Moreau and Vijay Dialani

• Related work:
  - UDDIM by Vijay Dialani (Southampton Univ)
  - UDDIg by John Colgrove (IBM)
Role of Registry

- Service Provider (Producer), Service Requestor (Consumer) and a Service Registry
- Service Provider (WSDL elements --> UDDI)
- Service Requestor (UDDI --> service name, service key)

- Concept is not new
  - CORBA: Name Service + Service Factory based on IDL
  - DCOM: Virtual reference table (GUID)
  - Jini: Lookup service (based on Java data types)
  - JXTA: XML documents (peer ID) -- collection of protocols for managing services
What do we need registries for?

• Publish Service Interface/Location/Access Info.
• Record Monitored Info
  - Performance data (APART, PYTHIA)
  - User info
  - Resource properties
  - Fault logs (archiving support)
• Who should publish and when
  - certificates or role-based security

• What should be done in a registry service and what elsewhere?
  - Bootstrapping (finding Factory, etc)
• How adaptive are the data structures to a different domain?
• Ease of use vs. representational complexity?
Web Services Composition

http://www.triana.co.uk/
http://www.gridoned.org/
http://www.gridlab.org/
UDDI

From Paul Freemantle (IBM, UK)
• Public Registries to be managed by institutions -- IBM, HP, etc
• Incentives for sharing are not obvious -- importance of “Data Custody”
• UDDI v3.0 -- X509 certificate support limit those who can publish

Gateway

Could act as a firewall
UDDI and eScience

Integrate different project descriptions/contact points for UK projects
UDDI and eScience

<table>
<thead>
<tr>
<th>recommended</th>
<th>principal investigator PI</th>
</tr>
</thead>
<tbody>
<tr>
<td>suggested</td>
<td>project manager PM</td>
</tr>
<tr>
<td></td>
<td>commercial manager</td>
</tr>
<tr>
<td></td>
<td>administrator</td>
</tr>
<tr>
<td></td>
<td>architect</td>
</tr>
<tr>
<td></td>
<td>technical authority chair</td>
</tr>
<tr>
<td></td>
<td>work package manager</td>
</tr>
<tr>
<td></td>
<td>steering group chair</td>
</tr>
<tr>
<td></td>
<td>programme board chair</td>
</tr>
<tr>
<td></td>
<td>Web or publicity manager</td>
</tr>
<tr>
<td></td>
<td>work package member</td>
</tr>
</tbody>
</table>

UK e-Science (e.g. GSC)

UBR (e.g. IBM)

registration

publisherAssertion

parent

child

e-Science project 1
e-Science project 2

peer

peer

UK e-Science All Hands Meeting 2003
UDDI Limitations

• Dealing with “missing” services -- and similarly, service documents which are out of date
• Search for services restricted to keyedReference or tModel -- this is quite restrictive
• Cannot find “partial” matches to services
• No support for provenance -- audit trails or logging capability
• Support for content sharing between a collection of individuals or companies -- and not via public UDDI Business Registry (UBR) operator nodes
• Ownership of “top most” registries (similar to root servers in Domain Name Servers)
UDDIe architecture

UDDIe request

Response

Servlet

SOAP Parser

Checks

Lease Manager

Java Classes

Database

• Syntax check
• Lease check
• Service name/type check

UK e-Science All Hands Meeting 2003
UDDIe structure

- **businessEntity** – The top-level XML element (includes support for "yellow pages" taxonomies)

- **businessService** – contains descriptive business service information about a group of related technical services including the group name, a brief description, technical service description information, service properties, service leasing details and category information.

- **bindingTemplate** – contains data relevant for applications that need to invoke or bind to a specific Web Service.

- **tModel** – Descriptions of specifications (protocols, formats etc) for Web Services or taxonomies; its role is to represent the technical specification of the Web Service, making it easier for Web Service consumers to find Web Services that are compatible with a particular technical specification.
Service Information

- **White pages**
  Basic contact information and identifiers about a company, including business name, address, contact information and unique identifiers such as D-U-N-S numbers or tax IDs. *Can we have something similar to encode references and ontologies for specific services?*

- **Yellow pages**
  Information that describes a web service using different categorizations (taxonomies) -- manufacturing, flowers or car sales.

- **Green pages**
  Technical information about Web Services that are exposed by a business (references to specifications or interfaces for Web Services, as well as support for pointers to various file and URL-based discovery mechanisms).

- **Blue pages *(UDDIe only)***
  Information that describes the service known as the service properties. This information allows other to discover web services based upon its properties (such as Quality of Service attributes).
UDDIe properties

- Service Leasing
  - register services with UDDI for a limited time period
- Query and advertisement of service based on service properties
  - user defined properties within a “propertyBag”
  - properties can be a set of parameters, methods available, sub-routine calls available etc
- Extend the find_service method
  - enable range based and logical (AND/OR) queries
- Can access UDDI and UDDIe in the same way
Service Leasing

• Services published for limited time periods
• Deals with services which change often or missing services

• Finite Lease:
  - Must define exact time periods for which service may be discoverable
  - Maximum duration defined by UDDIe admin.

• Infinite Lease:
  - Service made available as persistent services
  - Based on a ratio of finite/infinite leases -- defined by the UDDIe admin.
  - No guarantees of infinite leases

• By altering (1) lease duration, (2) finite/infinite leases, an admin can alter discovery performance
Service Leasing ... 2

- **Future Lease**
  - Allow lease period to start at a future point in time (i.e. service discoverable only after this time period)

- **Immediate Lease**
  - standard with UDDI -- service discoverable immediately on publication
Lease Manager

- Checks registry at periodic intervals
  - responsible for ensuring leases obtained for duration (or multiples) specified by admin.
  - Services request lease based on this
- Checks services based on service expiry times
- Lease manager is independent of database/registry
PropertyBag

• List of service properties that may be searched

Additional find qualifiers
• exactPropertyMatch (services which have exactly the requested properties)
• exactMatch (also uses keyedReference and categoryBag)
PropertyBag

Figure 5.1 The relations in the businessService entity
APIs

Inquiry APIs

find_binding  find_business
find_relatedBusinesses
find_service  find_tModel
get_bindingDetail
get_businessDetail
get_businessDetailExt
get_serviceDetail
get_tModelDetail

Publishing APIs

get_authToken
discard_authToken
delete_binding
delete_business
delete_publisherAssertion
delete_service
delete_tModel
get_assertionStatusReport
get_publisherAssertions
get_registeredInfo
save_binding  save_business
save_service  save_tModel
renew_lease
set_publisherAssertions
start_lease_manager
<find_service businessKey="*****" generic="2.0"
xmlns="urn:uddi.org:api_v2">
  <name>*****</name>
  <categoryBag>
    <keyedReference tModelKey="*****" keyName="*****"
keyValue="*****"/>
  </categoryBag>
  <tModelBag>
    <tModelKey>*****</tModelKey>
  </tModelBag>
  <propertyBag>
    <property>
      <propertyFindQualifier>*****</propertyFindQualifier>
      <propertyName>*****</propertyName>
      <propertyType>*****</propertyType>
      <propertyValue>*****</propertyValue>
    </property>
  </propertyBag>
</find_service>

Used for range based search
Property Find Qualifiers

- GREATER_THAN
- GREATER_OR_EQUAL
- LESS_THAN_OR_EQUAL
- EQUAL_TO
- NOT_EQUAL_TO
- AND/OR
Logical AND/OR Search

- FOR EACH element in the find_service message DO
  - Fetch the services which match with the element value
  - Add the services’ keys into the element result set
  - Add the element result set into total result set
- End For Loop

- IF Logical OR is required THEN
  - final result set = Union all element result set in the total result set
- ELSE IF
  - final result set = Intersect all element result set in the total result set
- END IF

- **element result set**: A set which contains all the service keys which matched the value of the element.
- **total result set**: A set which contains all the element sets
- **final result set**: The final result set which contains the wanted result.
Example: Logical AND/OR search

- `<find_service generic="2.0" xmlns="urn:uddi-org:api_v2">`
- `<name> X</name>`
- `<categoryBag> Y </categoryBag>`
- `<propertyBag> Z </propertyBag>`
- `</find_service>`

- result for the operation on X, Y and Z say R(X), R(Y) and R(Z)
- **OR**: R(X) U R(Y) U R(Z)
- **AND**: R(X) ^ R(Y) ^ R(Z)
Error Handling

- Range checks
- Checking on the existence of a property
  - via a DispositionReport
    - `<dispositionReport xmlns="urn:uddi-org:api_v3" >`
    - `<result errno="***" >`
    - `<errInfo errCode="***" />
    - `</result>
    - `</dispositionReport>`
- Checks on duplicates for Properties
  - via Unique tag (on PropertyName and PropertyType)
- Lease checking (exceptions)
  - InvalidLeaseDateException (data quality check)
  - RenewalTimeExceededException (rule based)
  - InfinitLeaseOutOfBoundException (rule based)
Example: QoS in Grid Computing

Domain1

User1  User2

AQoS

NRM  RM

S1  S2  Sn

Domain2

User1  User2

AQoS

NRM  RM

S1  S2  Sn

RM  Resource manager
NRM  Network resource manager
AQoS  Application QoS management layer
S  Service

---

Uses SDE to encode collected info

UK e-Science All Hands Meeting 2003
SLA Specifications

- \( V = V_{(\text{application-QoS})} \cup V_{(\text{middleware-QoS})} \cup V_{(\text{network-QoS})} \)
- \( R(V) \): set of relationships to evaluate
- For \( V_{(\text{network-QoS})} \)
  - \( R1 \): \((\text{delay} < 150\text{ms})\)
  - \( R2 \): \((\text{bandwidth} > 512\text{Kbps})\)
  - Contract: \((R1 \&\& R2)\)
- **Contract** \((C)\) properties:
  - \( C \) is an atomic unit
  - Evaluated to either TRUE or FALSE
  - Consistent
- Each service defines its capabilities
  - via a WSDL document
  - publishes these properties into the UDDIe registry
<find_service generic="2.0" xmlns="urn:uddi-org:api_v2">
  <name>MathService</name>
  <propertyBag>
    <property>
      <propertyFindQualifier>equal_to</propertyFindQualifier>
      <propertyName>cpu_count</propertyName>
      <propertyType>number</propertyType>
      <propertyValue>100</propertyValue>
    </property>
    <property>
      <propertyFindQualifier>equal_to</propertyFindQualifier>
      <propertyName>disk_storage</propertyName>
      <propertyType>number</propertyType>
      <propertyValue>150</propertyValue>
    </property>
  </propertyBag>
</find_service>
<wsdl:message name="getSumResponse">
  <wsdl:part name="return" type="SOAP-ENC:string"/>
</wsdl:message>
<wsdl:message name="getSumRequest">
  <wsdl:part name="symbol" type="SOAP-ENC:string"/>
</wsdl:message>

<wsdl:portType name="MathService">
  <wsdl:operation name="getSum" parameterOrder="symbol">
    <wsdl:input message="intf:getSumRequest"/>
    <wsdl:output message="intf:getSumResponse"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="mathSoapBinding" type="intf:MathService">
  <wsdlsoap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="getSum">
    <wsdlsoap:operation soapAction="" style="rpc"/>
    <wsdl:input>
      <wsdlsoap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://MathService-Interface" use="encoded"/>
    </wsdl:input>
    <wsdl:output>
      <wsdlsoap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://MathService-Interface" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>

<wSDL:QoS>
  <cpu_count>100</cpu_count>
  <disk_storage>150</disk_storage>
</wSDL:QoS>
<wSDL:definitions>
<save_service generic="2.0" xmlns="urn:uddie-org:api_v2">
  <name xml:lang="en">MathService</name>
  <bindingTemplates>
    <bindingTemplate bindingKey="">
      <accessPoint URLType="http">http://localhost/MathService/services/mathService</accessPoint>
      <tModelInstanceDetails>
        <tModelInstanceInfo tModelKey="*****">
          <instanceDetails>
            <overviewDoc>
              <overviewURL>http://localhost/MathService/MathService_interface.wsdl</overviewURL>
            </overviewDoc>
          </instanceDetails>
        </tModelInstanceInfo>
      </tModelInstanceDetails>
    </bindingTemplate>
  </bindingTemplates>
  <propertyBag>
    <property>
      <propertyName>cpu_count</propertyName>
      <propertyType>number</propertyType>
      <propertyValue>100</propertyValue>
    </property>
    <property>
      <propertyName>disk_storage</propertyName>
      <propertyType>number</propertyType>
      <propertyValue>150</propertyValue>
    </property>
  </propertyBag>
</businessService> </save_service>
Output obtained from UDDIe registry

WIZARD to publish properties into UDDIe
Conclusion

• Implementation of UDDIE
• Primary contribution by Ali Shaikhali -- available from:
  http://www.cs.cf.ac.uk/User/A.Shaikhali/uddie/
• Tested with various databases: Oracle, DB2, Access, mySQL
• Aim to keep compatibility with UDDI
• Primary interest in associating Quality of Service with Web/Grid Services
Extra Stuff
UDDI Update

• Grid related requirements for UDDI by Colgrave and Dovey
  - Differences between Science and Business services
• OASIS UDDI Specification Technical Committee managed
• Queries can now be based on:
  - finding tmodel based on a WSDL portType
  - find all registered bindings of a portType
  - find registered implementations of a binding
  - find all registered implementations of a portType
• Any logical/physical WSDL structure supported
UDDI Update ... 2

- **WSDL <-> UDDI**
  - Service <-> BusinessService
  - Port <-> bindingTemplate
  - binding <-> binding tModel
  - portType <-> portType tModel
- **portType(s) <-> portType tmodel(s)**
- **UDDI for eScience**
  - [http://esc.dl.ac.uk/Papers/UDDI/uddi.pdf](http://esc.dl.ac.uk/Papers/UDDI/uddi.pdf)
- **WSDL 1.2 -- portType inheritance support**
- **gWSDL == WSDL extensions for OGSI**
UDDI-aware ServiceGroup

- UDDI-based Registry for appropriate Grid Service instances
  - Long-lived
  - Accessible to multiple users
  - For example, other ServiceGroups, Factories
- Register sufficient information to publish to UDDI in accordance with enhanced WSDL
  Technical Note
  - Generic UDDI portType?
  - UDDI-specific extensions to the three ServiceGroup-related portTypes?
UDDI-aware ServiceGroup

- ServiceGroupRegistration: add results in publication to UDDI
- Lifetime of ServiceGroupEntry: mapped to publication/removal from UDDI
- UDDI-aware HandleResolver: