5-21-2007

Using SAWSDL for Semantic Service Interoperability

Kunal Verma
Wright State University - Main Campus

Amit P Sheth
Wright State University - Main Campus, amit.sheth@wright.edu

Follow this and additional works at: http://corescholar.libraries.wright.edu/knoesis

Part of the Bioinformatics Commons, Communication Technology and New Media Commons, Databases and Information Systems Commons, OS and Networks Commons, and the Science and Technology Studies Commons

Repository Citation

This Presentation is brought to you for free and open access by the The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis) at CORE Scholar. It has been accepted for inclusion in Kno.e.sis Publications by an authorized administrator of CORE Scholar. For more information, please contact corescholar@www.libraries.wright.edu.
Using SAWSDL for Semantic Service Interoperability

Kunal Verma
Accenture Technology Labs
k.verma@accenture.com

Amit Sheth,
Karthik Gomadam
Kno.e.sis Center, Wright State University
http://knoesis.wright.edu
Introduction

- Semantic Annotations for WSDL (SAWSDL) is W3C Candidate Recommendation
- It defines a mechanism to add semantic annotations to Web services
- It is based on W3C member submission WSDL-S
  - WSDL-S was proposed by University of Georgia METEOR-S Team* and IBM
- It is an important first step by W3C to add support for semantic modeling the Web service stack.

Now mainly at Kno.e.sis Center, Wright State University
Details about SAWSDL

- **Standard Activity**
  - W3C SAWSDL Working group
    - [http://www.w3.org/2002/ws/sawsdl/](http://www.w3.org/2002/ws/sawsdl/)
  - W3C WSDL-S member submission Web page
    - [http://www.w3.org/Submission/WSDL-S/](http://www.w3.org/Submission/WSDL-S/)

- **Tools**
  - SAWSDL4J by Wright State University (Dayton, OH) and University of Georgia (UGA, Athens, GA)
  - Radiant: WSDL-S/SAWSDL Annotation Tool by University of Georgia
  - Semantic Tools for Web Services by IBM alphaWorks
  - WSMO Studio by DERI

- **Some Relevant Papers**
Part 1

BACKGROUND AND MOTIVATION
Outline

- Part 1 – Background and Motivation
  - Evolution of SOA
    - Value proposition
    - Web Services Stack
    - Web Service Description Language
  - Adding Semantics to SOA
    - Motivation
    - Value proposition
    - How will Semantics Change SOA

- Part 2: Deep Dive into SAWSDL
- Part 3: Using SAWSDL
- Part 4: SAWSDL Tools
Semantic Oriented Architecture (SOA)

- SOA has transitioned from a vision to commonly implemented architecture

- Web services are a set of XML based standards
  - Front-runners to implement SOA based solutions
  - Widely supported by a number of vendors
Value Proposition of SOA and Web Services

- Interoperability
  - Ability to connect applications across platforms, languages, operating systems, geographic regions

- Re-use
  - Ability to re-use services across applications

- Insight
  - Direct visibility into business processes. Ability to link business goals to business processes

- Agility
  - Ability to react, ability to quickly create new processes
Web Service Standards

- WS-BPEL: Composition
- WSDL, WS-Policy, UDDI: Description
- WS-Security
- WS-Reliable Messaging
- WS-Transaction: Quality of Service
- SOAP (Logical Messaging)
- XML, Encoding
- Other protocols
- Other services: Interaction
What is WSDL

- An extensible, platform independent XML language for “describing” services.

- Provides functional description of Web services:
  - IDL description
  - Access protocol and deployment details
  - All of the functional information needed to programmatically access a service, contained within a machine-readable format

- Does not include
  - QoS
  - Taxonomies
  - Business information

- WSDL is a component definition language for Web service component
WSDL Example

<wsdl:description targetNamespace="http://www.w3.org/2002/ws/sawSDL/spec/wsdl/order#"
 xmlns:wsdl="http://www.w3.org/ns/wsdl" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <xs:element name="processPurchaseOrderResponse" type="xs:string">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="orderConfirmation" type="xs:integer" />
          <xs:element name="deliveryDate" type="xs:integer" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </wsdl:types>
  <wsdl:interface name="PurchaseOrder">
    <wsdl:operation name="order" pattern=wsdl:in-out>
      <input messageLabel="processPurchaseOrderRequest" element="tns:processPurchaseOrderRequest"/>
      <output messageLabel="processPurchaseOrderResponse" element="processPurchaseOrderResponse"/>
    </wsdl:operation>
    <wsdl:operation name="cancel" pattern=wsdl:in-out>
      <input messageLabel="processCancelRequest" element="tns:processCancelRequest"/>
      <output messageLabel="processCancelResponse" element="processCancelResponse"/>
    </wsdl:operation>
  </wsdl:interface>
</wsdl:description>
ADDING SEMANTICS TO SOA
Semantic Web Services

- Rich research history—too much to review here
- SWS related submissions to W3C
  - OWL-S: http://www.w3.org/Submission/OWL-S/
  - WSMO: http://www.w3.org/Submission/2005/06/
  - SWSF: http://www.w3.org/Submission/SWSF/
  - WSDL-S: http://www.w3.org/Submission/WSDL-S/
- W3C Workshop at Innsbruck, leading to community agreement to focus on limited scope and evolutionary approach championed by WSDL-S, leading to SAWSDL WG
Semantics to Web Services: The ingredients

- Conceptual Model/Ontology
  - An agreed upon model that captures the semantics of domain
  - Common Nomenclature
  - Domain Knowledge (facts)

- XML based service description
  - Standards and specifications like WSDL for web service description, WS-Agreement for capturing agreements etc.

- Annotate the service description
What does Semantics bring to the table?

- Better Reuse
  - Semantic descriptions of services to help find relevant services

- Better Interoperability
  - Beyond syntax to semantics, mapping of data exchanged between the services (very time consuming without semantics, just as XML in WSDL gives syntactic interoperability, SAWSDL gives semantic interoperability)

- Configuration/Composition
  - Enable dynamic binding of partners

- Some degree of automation across process lifecycle
  - Process Configuration (Discovery and Constraint analysis)
  - Process Execution (Addressing run time heterogeneities like data heterogeneities.)
SAWSDL

- Offer an evolutionary and compatible upgrade of existing Web services standards

- Externalize the semantic domain models
  - agnostic to ontology representation languages (although W3C recommended RDFS or OWL are likely to be often used)
  - reuse of existing domain models (in some domains, usable ontologies have been built, eg life science and health care)
  - allows annotation using multiple ontologies (same or different domain)

- Updating tools around WSDL is relatively easier
Guiding principles...

- Support semantic annotation of Web Services whose data types are described in XML schema.
- Provide support for rich mapping mechanisms between Web Service schema types and ontologies.
Why use SAWSDL

- Build on existing Web Services standards using only extensibility elements
- Mechanism independent of the semantic representation language (though OWL is supported well)
- SAWSDL provides an elegant solution
  - Help integration by providing mapping to agreed upon domain models (ontologies, standards like Rosetta Net, ebXML)
  - Better documentation by adding functional annotation
- Ease in tool upgrades
  - e.g. wsif / axis invocation
Part 2

DEEP DIVE INTO SAWSDL
Outline

- Part 1 – Background and Motivation
- Part 2: Deep Dive into SAWSDL
  - SAWSDL Scope
  - The extensibility attributes
  - Annotating operations
  - Annotating faults
  - Annotating types
  - Annotating interfaces
  - SAWSDL Example
- Part 3: Using SAWSDL
- Part 4: SAWSDL Tools
SAWSDL Scope

No SAWSDL annotations defined for these WSDL components

Annotated using modelReference

Annotated using modelReference and schemaMapping

Picture from: http://www.w3.org/TR/wsdl20-primer/
SAWSDL at a glance

Semantics:
- ontology classes
  - discovery, composition
  - filtering, ranking
- lifting/lowering mappings
  - mediation, invocation
- functionality categories
  - publishing, discovery, composition
- anything, really

Ack: Jacek Kopecky
SAWSDL defines two extensibility attributes

- **modelReference:** This can be used to specify the association between a WSDL or XML Schema component and a concept in some semantic model.
  - It can be used to annotate the following:
    - WSDL components
      - Interfaces
      - Operations
      - faults
    - WSDL Type Definitions
      - XML Schema complex type definitions
      - Simple type definitions
      - element declarations
      - attribute declarations

- **liftingSchemaMapping:** This can be used to specify mappings between WSDL Type Definitions in XML and semantic data.

- **loweringSchemaMapping:** This can be used to specify mappings between semantic data and WSDL Type Definitions in XML.
Using modelReference to annotate operations

The annotation of the operation element carries a reference to a concept in a semantic model that provides a high level description of the operation, specifies its behavioral aspects or includes other semantic definitions.
Using modelReference to annotate faults

The annotation of the fault element carries a reference to a concept in a semantic model that provides a high level description of the fault and can include other semantic definitions.
Annotating Types

- Following WSDL Type Definitions can be annotated using the `modelReference`, `liftingSchemaMapping` and `loweringSchemaMapping` extension attributes
  - XML Schema complex type definitions
    - Bottom-level annotation
    - Top level annotation
  - Simple type definitions
  - element declarations
  - attribute declarations
Annotating types

1. `modelReference` to establish a semantic association
2. `liftingSchemaMapping` and `loweringSchemaMapping` to provide mappings between XML and semantic model

WSDL complex type element

```
<wSDL:types>
  (...)
  <complexType name="Address">
    <sequence>
      <element name="StreetAd1" type="xsd:string"/>
      <element name="StreetAd2" type="xsd:string"/>
    </sequence>
  </complexType>
  (...)
</wSDL:types>
```
Annotating complex types with modelReference – Bottom level annotation

```xml
<complexType name="POItem">
  <all>
    <element name="dueDate" nillable="true" type="dateTime"
      sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#DueDate"/>
    <element name="qty" type="float"
      sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#Quantity"/>
    <element name="EANCode" nillable="true" type="string"
    <element name="itemDesc" nillable="true" type="string"
  </all>
</complexType>
```

OWL ontology

```
Item
   ↓
  hasDueDate
  ↓
 dueDate

ItemDesc
   ↓
  hasItemDesc

Quantity
   ↓
  hasQuantity
```

WSDL complex type element
Annotating complex types with modelReference – Top level annotation

```xml
<complexType name="POItem"
sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#DueDate">
  <all>
    <element name="dueDate" nillable="true" type="dateTime" />
    <element name="qty" type="float"/>
    <element name="EANCode" nillable="true" type="string"/>
    <element name="itemDesc" nillable="true" type="string"/>
  </all>
</complexType>
```

WSDL complex type element

**OWL ontology**
Using schemaMapping with modelReference

- Any mapping language can be used for liftingSchemaMapping
  - Recommended languages: XSLT, Xquery
- Any mapping language can be used for liftingSchemaMapping
  - Recommended languages: SPARQL to query ontology, followed by XSLT, Xquery

```xml
<complexType name="POAddress"
    sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#Address"
    sawsdl:loweringSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapping/Ont2POAddress.xslt">
    <all>
        <element name="streetAddress" type="xsd:string" />
        <element name="poBox" type="xsd:string" />
        <element name="city" type="xsd:string" />
        <element name="zipCode" type="xsd:string" />
        <element name="state" type="xsd:string" />
        <element name="country" type="xsd:string" />
        <element name="recipientInstName" type="xsd:string" />
    </all>
</complexType>
```

OWL ontology

WSDL complex type element
liftingSchemaMapping example using XSLT

```xml
<xsl:transform version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:po="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
    xmlns:POOntology="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#">
  <xsl:output method="xml" version="1.0" encoding="iso-8859-1" indent="yes" />
  <xsl:template match="/">
    </POOntology:OrderConfirmation>
    <POOntology:Address rdf:ID="Address1">
      <POOntology:has_StreetAddress rdf:datatype="xs:string">
        <xsl:value-of-select="concat(POAddress/streetAddress)"/>
      </POOntology:has_StreetAddress >
      <POOntology:has_City rdf:datatype="xs:string">
        <xsl:value-of select="POAddress/city"/>
      </POOntology:has_City>
      <POOntology:has_Zip rdf:datatype="xs:string">
        <xsl:value-of select="POAddress/zip"/>
      </POOntology:has_State>
    </xsl:template>
</xsl:transform>
```
Using schemaMapping with modelReference (heterogeneity)

```xml
<complexType name="POAddress"
  sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#Address"
  sawsdl:loweringSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapping/Ont2POAddress.xslt">
  <all>
    <element name="streetAddr1" type="xsd:string" />
    <element name="streetAddr2" type="xsd:string" />
    <element name="poBox" type="xsd:string" />
    <element name="city" type="xsd:string" />
    <element name="zipCode" type="xsd:string" />
    <element name="state" type="xsd:string" />
    <element name="country" type="xsd:string" />
    <element name="recipientInstName" type="xsd:string" />
  </all>
</complexType>
```

Data level heterogeneity

OWL ontology

WSDL complex type element
liftingSchemaMapping example using XSLT (heterogeneity)

```xml
<xsl:transform version="2.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:po="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
 xmlns:POOntology="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#">
 <xsl:output method="xml" version="1.0" encoding="iso-8859-1" indent="yes"/>
 <xsl:template match="/"
</POOntology:OrderConfirmation>
 <POOntology:Address rdf:ID="Address1">
  <POOntology:has_StreetAddress rdf:datatype="xs:string">
   <xsl:value-of-select="concat(POAddress/streetAddr1,POAddress/streetAddr2)"/>
  </POOntology:has_StreetAddress>
  <POOntology:has_City rdf:datatype="xs:string">
   <xsl:value-of select="POAddress/city"/>
  </POOntology:has_City>
  ............
 </xsl:template>
</xsl:transform>
```
A `modelReference` on a WSDL `interface` element provides a reference to a concept or concepts in a semantic model that describe the Interface.
<wsdl:description targetNamespace="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
xmlns:wsdl="http://www.w3.org/ns/wsdl" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sawsdl="http://www.w3.org/ns/sawsdl">
<wsdl:types>
  <xs:element name="processPurchaseOrderResponse" type="xs:string
   sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/rosetta#PurchaseOrderResponse"
  ......
  </xs:element>
</wsdl:types>
<interface name="PurchaseOrder"
  sawsdl:modelReference="http://example.org/categorization/products/electronics />
<operation name="order" pattern=wsdl:in-out
  sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/rosetta#RequestPurchaseOrder">
  <input messageLabel = "processPurchaseOrderRequest"
       element="tns:processPurchaseOrderRequest"/>
  <output messageLabel ="processPurchaseOrderResponse"
       element="processPurchaseOrderResponse"/>
</operation>
<operation name="cancel" pattern=wsdl:in-out
  sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/rosetta#CancelOrder">
  <input messageLabel = "processCancelRequest"
       element="tns:processCancelRequest"/>
  <output messageLabel ="processCancelResponse"
       element="processCancelResponse"/>
</operation>
</interface>
</wsdl:description>
Part 3

USING SAWSDL
Outline

- Part 1 – Background and Motivation
- Part 2: Deep Dive into SAWSDL
- Part 3: Using SAWSDL
  - Using SAWSDL with UDDI for Discovery
  - Using SAWSDL with WS-BPEL for dynamic binding
  - Using SAWSDL with Apache Axis for Data Mediation
- Part 4: SAWSDL Tools
Part 3.1

USING SAWSDDL WITH UDDI FOR DISCOVERY
Using SAWSDL with UDDI

- The semantic annotations in SAWSDL represent the semantic signature of a service
  - Using annotations from semantic models can help with discovering services with certain semantic signatures

- As part of METEOR-S project, we investigated publishing WSDL-S/SAWSDL files in UDDI Registries [1]

- Builds upon following previous discovery implementations
  - Extended matching presented in [2] to consider operations and service level metadata
  - Extends the approach presented “WSDL to UDDI Mapping” [3] to get SAWSDL to UDDI Mapping


1. SAWSDL file creating using annotations (modelReferences) pointing to semantic model

2. Service published in UDDI along with annotations

3. Service request created using terms from semantic model

4. Discovery based on annotations
Using SAWSDL to represent semantic templates

- **Semantic Templates** capture the functionality of a Web service with the help of ontologies/other domain models.
- Find a service that sells RAM in Athens, GA. It must allow the user to return and cancel, if needed.
- Semantic templates can also have non-functional (QoS) requirements such as response time, security, etc. using WS-Policy.

**Semantics Templates represented using SAWSDL without implementation details**
Basic UDDI Data Structures

- **businessEntity**: Information about the party who publishes information about a service.

- **businessService**: Descriptive information about a particular family of technical services.

- **bindingTemplate**: Technical information about a service entry point and construction specifications.

- **tModel**: Descriptions of specifications for services or taxonomies. Basis for technical fingerprints.

bindingTemplate data contains references to tModels. These references designate the interface specifications for a service.

Mapping SAWSDL to UDDI Data Structures for publication
Sample query using UDDI API

```xml
<find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
  <categoryBag>
    <keyedReference tModelKey="WSDL_TYPE_T_MODEL_KEY"
      keyName="WSDL type"
      keyValue="operation"/>
    <keyedReference tModelKey="OPERATION_MODELREF_TMODEL_KEY"
      keyName="Operation-modelReference"
      keyValue="http://example.org/rosetta#RequestPurchaseOrder"/>
    <keyedReference tModelKey="INPUT_MODELREF_TMODEL_KEY"
      keyName="Input-modelReference"
      keyValue="http://example.org/rosetta#PurchaseOrderRequest"/>
    <keyedReference tModelKey="OUTPUT_TMODEL_KEY"
      keyName="Output-modelReference"
      keyValue="http://example.org/rosetta#PurchaseOrderConfirmation"/>
  </categoryBag>
</find_tModel>
```
Part 3.2

USING SAWSDL WITH WS-BPEL FOR DYNAMIC BINDING
Sample BPEL Process with WSDL services

Part 1
Supplier Service

quote
order

Partner1 Web Service (WSDL)

Part 2
Supplier Service2

quote
order

Partner1 Web Service (WSDL)

BPEL Process

ReceiveOrder

Partner1: getQuote

Partner2: getQuote

Analyze Quotes

Partner1: order

Partner2: order

Reply
Sample BPEL Process with SAWSDL services

Part 1
Supplier Service

Partner1 Web Service (SAWSDL)

quote
order

Part 1 Web Service (SAWSDL)

Part 2
Supplier Service2

Partner1 Web Service (SAWSDL)

quote
order

WSDL files can be seamlessly replaced by SAWSDL files

BPEL Process

ReceiveOrder

Analyze Quotes

Partner1: getQuote

Partner2: getQuote

Partner1: order

Partner2: order

Reply
Advantages of using SAWSDL with WS-BPEL

- Can link actual SAWSDL based Web services or semantic templates.
- SAWSDL based Web services can be used to replace existing services in case of business or physical failures.
- SAWSDL based semantic templates can be used for dynamic binding:
  - They can be replaced with actual services either at run-time or deployment time to achieve dynamic binding.
- BPEL already has support for dynamic binding. SAWSDL provides a mechanism to represent functional semantics of a service, that can be used for discovery.

Kunal Verma, Configuration and Adaptation of Semantic Web Processes, Ph.D. Thesis, Department of Computer Science, The University of Georgia, August 2006
Sample BPEL Process with SAWSDL services

Advantages:
- Services can be added or deleted by publishing them to the registry. Process doesn’t have to be changed and re-deployed
- Process can be re-used for different types of orders by changing service discovery criteria
Part 3.3

USING SAWSDL FOR DATA MEDIATION
SAWSDL support for data mediation

- User specified mappings from Web service message element to semantic model concept (say OWL Ontology)
  - *liftingSchemaMapping*: from WS message element to OWL concept
  - *loweringSchemaMapping*: from OWL concept to WS message element

```xml
<complexType name="Address">
  <sequence>
    <element name="StreetAd1" type="xsd:string"/>
    <element name="StreetAd2" type="xsd:string"/>
    ..........  
  </sequence>
</complexType>
```
Realizing data mediation

- Web services interoperate by re-using these mappings.
  - Ontologies now a vehicle for Web services to resolve message level heterogeneities
Prototype implemented using METEOR-S

- **METEOR-S Middleware**
  - **EPR handler – End Point Resolution handler**
    - For clients to use the middleware
    - Reroute SOAP messages to middleware
  - **DM handler – Data Mediation handler**
    - Main component for facilitating data mediation
    - Works with the EPR handler + a mapping processing engine (SAXON for XQuery / XSLT)

DM Handler – a closer look

- Each time a Web service is invoked
  - obtains the ‘liftingSchemaMapping‘ and loweringschemaMapping‘ functions from SAWSDL locations (using the SAWSDL4J API)
  - performs the lifting and lowering transformations on the incoming SOAP message using a mapping processor/engine (SAXON for XQuery and XSLT)
  - updates the SOAP message. Appropriate Axis handlers then invoke the Web service with the transformed message.

Note: This implementation used WSDL-S, so it used upcast and downcast instead of liftingSchemaMapping‘ and loweringschemaMapping‘
SAWSDL4J API available at: http://knoesis.wright.edu/opensource/sawssl4j/
Walk through example – WS1 invocation

Conceptual Model

XML (SOAP B) TO OWL

SAXON Xquery/ XSLT Engine

DM Handler

* Enlarged view of the DM Handler

OWL TO XML (SOAP C)

SOAP C

SOAP message

Modified by DM handler

SOAP B

Modified SOAP message

with actual Web service EPR

SOAP A

Original SOAP message

AXIS 2.0

User phases

* DM handler

EPR handler

Axis phases

User phases

* DM handler

EPR handler

InFlow

OutFlow

METEOR-S Middleware

1,2 a SOAP A

1,2 b SOAP B

1c SOAP C

1,2 d

Web service 1

1,2 e

Web service 2

CLIENT WS1 -> WS2

SOAP A

Original SOAP message

SOAP B

Modified SOAP message

with actual Web service EPR

SOAP C

SOAP message

Modified by DM handler
Part 4

SAWSDL TOOLS
Outline

- Part 1 – Background and Motivation
- Part 2: Deep Dive into SAWSDL
- Part 3: Using SAWSDL
- Part 4: SAWSDL Tools
  - METEOR-S Tools
  - IBM Tools
  - DERI Tools
A number of tools are available from the METEOR-S project at the University of Georgia and Wright State University:

- **SAWSDL4J**: API for manipulating SAWSDL files. Nightly builds available for download at http://knoesis.wright.edu/opensource/sawsdl4j
- **Semantic Services Registry**: A distributed registry with SOAP and REST bindings for publishing, versioning and discovering SAWSDL services. Nightly builds will be available in Fall 2007.
- **Axis2 Plugins for Dynamic Binding**: Plugins to Apache Axis 2 to support dynamic binding of services. Nightly builds will be available in late Fall 2007.

**DEMO/OVERVIEW**
Semantic Annotation and Publication - Radiant
Semantic Web Services Discovery: Lumina
IBM Tools

- A number of tools from IBM are available for WSDL-S/SAWSDL at http://www.alphaworks.ibm.com/tech/wssem

- Tools are available as part of the ETTK toolkit for the following features:
  - Web Service Interface Matching
  - Web Service Discovery
  - Web Service Composition

- DEMO/OVERVIEW
DERI Tools

- Integrated with WSMO Studio
  - Support for all modelRef annotations
  - Limited lifting/lowering schema support
    - No support for XSLT / SPARQL mapping definitions
  - Support for WSDL 1.1

- User Interface
  - Tree view of the WSDL structure
  - Text view with syntax highlighting
  - Online demo at http://www.wsmostudio.org/demo/sawssdl.htm

- DEMO/OVERVIEW
WSMO Studio

- SWS Modelling Environment for WSMO
  - Java & Eclipse based
  - Open source (LGPL)
  - http://www.wsmostudio.org

- Components
  - WSMO editor
  - Choreography designer
  - SAWSDL editor

- Features
  - Import/export (WSML / RDF / OWL-DL)
  - Integrated WSML reasoners (Pellet, MINS, KAON2)
  - Front-end to SWS repositories (ORDI, IRS-III, WSMX)
  - Front-end to SWS matchmakers (EPFL)
Conclusions & Next Steps

- SAWSDL on its way to be a W3C recommendation

- SAWSDL is being evaluated by a number of businesses.

- WSDL-S idea of semantic annotation is leading to several other annotations in SOA: SAREST (being developed at Wright State University & UGA), semantic annotation of policy descriptions, etc.

- Some additional work being taken up by W3C Incubator on SWS-testbed http://www.w3.org/2005/Incubator/swsc/
Acknowledgements

- We would like to thank the following individuals for providing slides for this tutorial:
  - Rama Akkiraju, IBM T.J. Watson Research Center (WSDL-S team and W3C SAWSDL Committee Member)
  - Jacek Kopecky, DERI, (W3C SAWSDL Committee Chair)
  - Marin Dimitrov, DERI
  - John Miller, UGA (WSDL-S team and SAWSDL committee member)
  - Meenakshi Nagarajan, Kno.e.sis. Center, Wright State University (WSDL-S team member)