Semantic Web Services
SBBD 2008

Renato Fileto
Frank Siqueira

Collaborators (SWS tools selection):
Giorgio Merize
Rodrigo Zeferino

{fileto|frank|giorgio|rodrigo.zeferino}@inf.ufsc.br
INE/CTC/UFSC

Topics

- **Introduction**
- Web Services (WS)
- Semantic Web (SW)
- Semantic Web Services (SWS)
- Some Major Efforts towards SWS
  - WSDL-S
  - OWL-S
  - SWSF (SWSO + SWSL)
  - WSMO (WSMO + WSML + WSMX)
- **Software Tools:** WSMT, WSMX, IRS-III, ...
- **Case study:** Travelling to SBBD
Introduction

Web Services Technology
(discovery, selection, composition, and web-based execution of services)

+ Semantic Web
(ontologies and machine supported data interpretation)

= Semantic Web Services
(integrated solution for realizing the vision of the next generation of the Web)

The Web

- The Web was initially designed for application to human interactions
- Served very well its purpose:
  ◆ Information sharing: a distributed content library.
  ◆ Enabled B2C e-commerce.
  ◆ Non-automated B2B interactions.
- How did it happen?
  ◆ Built on standards: HTTP, HTML, URI, ...
  ◆ Very few assumptions made about computing platforms.
  ◆ Ubiquity.
What’s next?

- The Web is everywhere. There is a lot more we can do!
  - E-marketplaces.
  - Open, automated B2B e-commerce.
  - Business process integration on the Web.
  - Resource sharing, distributed computing.

- Current approach is *ad-hoc* on top of existing standards.
  - e.g., application-to-application interactions with HTML forms.

- **Goal:** enabling systematic and automated application-to-application interaction on the Web.

W3C’s Protocol Working Group

“...the Web can grow significantly in power and scope if it is extended to support [automated] communication between applications, from one program to another.”

W3C’s Protocol Woking Group
Topics

- Introduction
- **Web Services (WS)**
- Semantic Web (SW)
- Semantic Web Services (SWS)
- Some Major Efforts towards SWS
  - WSDL-S
  - OWL-S
  - SWSF (SWSO + SWSL)
  - WSMO (WSMO + WSML + WSMX)
- **Software Tools:** WSMT, WSMX, IRS-III, ...
- **Case study:** Travelling to SBBD

Web Services

- Encapsulated, loosely coupled Web “components” that can bind dynamically to each other.

- Services are programmatically accessible over standard Internet protocols
A Web Service

- Identified by an URI
- Self-describing and openly accessible
- Can be remotely invoked through a well-defined interface
- Exchanges data in XML format
- Interacts with applications and other services via messages exchange (HTTP/SMTP)
- Independent from other services and applications, but can cooperate with them

Web Service Architecture

Based on the Service Oriented Architecture (SOA)
Obviously, there are other technologies for doing this

Web services standardize connections, enabling “plug and play” on the Web.

Web Service Objectives

- Universal interoperability
- Exploit ubiquity of the Web
- Enable dynamic binding
- Efficiently support open environment (Web) and more restrict environments if necessary
- Minimize incompatibility costs
  - programming languages,
  - operating systems,
  - network protocols.

An effort towards building a distributed computing platform on the Web.
Why Web Services?

- Based on generally accepted standards
- Require little additional infrastructure
- Loose coupling
- Focus in messages and documents, not *APIs*
- Easy to use
- Complement existing technologies
- Interoperability
- Everybody use, have plans to use or is forced to use

Technology Evolution

Granularity, Coupling & Abstraction

High Abstraction

Low Abstraction

Fine grained

Coarse grained

High Coupling

Low Coupling
Web Services Framework

- What goes “on the wire”: Formats and protocols.
- What describes what goes on the wire: Description languages.
- How to find the services we need: Discovery and selection of services.
- How to assemble and control the execution of services in processes on the Web: Composition of services.

Current Web Services Technologies

- Standards for publication, invocation & search
  - Unicode, URI + namespaces
  - XML (eXtensible Markup Language) + XML-Schema
  - SOAP (f.k.a. Simple Object Access Protocol)
  - WSDL (Web Services Definition Language)
  - UDDI (Unified Data Description and Interchange)

- Implementation technologies
  - .NET (Microsoft)
  - Java Technology for Web Services (SUN)
  - ... and many others.
Web Service Interaction

1. Registers Service
2. Service Discovery
3. Gets Service URI
4. Asks for Description
5. Gets WSDL
6. Invokes Service
7. Gets Response (optional)

Current Web Services Standards

- Cooperative Processes
- Semantic Web Services
- UDDI
- WSDL
- SOAP
- Internet Protocols (HTTP)
- XML + XML-Schema
- URI + NameSpaces
- Character Encoding (Unicode)

Access Control & Security Policies
Characters Encoding

- `<xml encoding='UTF-8'>`
- `<xml encoding='UTF-16'>`
- `<xml encoding='EUC-JP'>`
- `<xml version="1.0" encoding='ISO-8859-1'>`

URI (Uniform Resource Identifier)

An URI identifies an abstract or physical resource
URNs (Uniform Resource Names)
URLs (Uniform Resource Locators)

Examples:
ftp://ftp.is.co.za/rfc/rfc1808.txt
http://www.ietf.org/rfc/rfc2396.txt
mailto:John.Doe@example.com
news:comp.infosystems.www.servers.unix
ldap://[2001:db8::7]/c=GB?objectClass?one
telnet://192.0.2.16:80/
tel: +1-816-555-1212
NameSpaces

- `<x xmlns:edi="http://ecommerce.org/schema">`  
  <!-- the "edi" prefix is bound to http://ecommerce.org/schema for the "x" element and contents -->
  </x>

  `<title>Cheaper by the Dozen</title>`  
  `<isbn:number>1568491379</isbn:number>`
  </book>

- `<schema xmlns="http://www.w3.org/2001/XMLSchema">`  
  `<schema>`

XML – eXtensible Makup Language

- `<xml version="1.0"?>`  
  `<!DOCTYPE people SYSTEM "http://www.wsno.org/workinggroup.dtd">`  
  `<!−− This XML document gives information about working group members of the WSMO working group −−>`  
  `<people xmlns="http://www.wsno.org/namespace">`  
  `<title>WSMO working group members</title>`  
  `<member chair="yes">`  
    `<firstname>Dieter</firstname><lastname>Fensel</lastname>`  
    `<affiliation>DERI International</affiliation>`
  </member>`
  `<member chair="yes">`  
    `<firstname>John</firstname><lastname>Domingue</lastname>`  
    `<affiliation>Open University</affiliation>`
  </member>`
  `<member>`  
    `<firstname>Axel</firstname><lastname>Polleres</lastname>`  
    `<affiliation>Univ. Rey Juan Carlos</affiliation>`
  </member>`
  `⋯`
</people>`
**DTD**

```xml
<!DOCTYPE people [ 
<!ELEMENT people (title,member+)>
<!ELEMENT member (firstname,lastname,affiliation+)>
<!ATTLIST member chair (yes|no) "no"/>
<!ELEMENT title (#PCDATA)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT affiliation (#PCDATA)>
]>
```

**XML-Schema**

An *XML-Schema* document describes:
- elements,
- attributes,
- relationships,
- etc.

that can be used in one or more XML documents, i.e., defines a class of XML documents that follow a set of structural and data constraints

- *XML-Schema* uses the *XML* syntax
- *XML-Schema* is more robust, versatile and powerful than *DTD* (*Document Type Definition*)
XML-Schema (example)

```xml
<xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns="http://www.wsmo.org/namespace"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault="qualified"
 targetNamespace="http://www.wsmo.org/namespace">
 <xs:element name="people">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="firstname" type="namestring" minOccurs="1" maxOccurs="2"/>
       <xs:element name="lastname" type="namestring" minOccurs="1" maxOccurs="2"/>
       <xs:element name="affiliation" type="namestring" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:complexType>
 </xs:element>
</xs:schema>
```

XML-Schema (example cont)

```xml
<xs:complexType name="person">
 <xs:sequence>
   <xs:element name="firstname" type="namestring" minOccurs="1" maxOccurs="2"/>
   <xs:element name="lastname" type="namestring" minOccurs="1" maxOccurs="2"/>
   <xs:element name="affiliation" type="namestring" maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="chair" default="no">
   <xs:simpleType>
     <xs:restriction base="xs:string">
       <xs:enumeration value="yes"/>
       <xs:enumeration value="no"/>
     </xs:restriction>
   </xs:simpleType>
 </xs:attribute>
</xs:complexType>
```
XML-Schema (example cont)

```xml
<xs:simpleType name="namestring">
  <xs:restriction base="xs:string">
    <!-- This pattern says that names are strings starting with an uppercase letter -->
    <xs:pattern value="^[p]\{Lu\}\.*"/>
  </xs:restriction>
</xs:simpleType>
```

Validating XML Documents

- **Well-formed document**: satisfies the restrictions expressed in the XML specification (http://www.w3.org/TR/2004/REC-xml-20040204)

- **Valid document**: satisfies the restrictions expressed in a schema specification (valid elements, attributes, nesting, etc.) written in DTD or XSL associated to the XML document
Request example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body xmlns:ns1="http://ufsc.br/previsao">
        <ns1:getMinTemperature>
            <location>Florianópolis</location>
        </ns1:getMinTemperature>
    </S:Body>
</S:Envelope>
```
**SOAP**

- **Return example**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns1:getMinTemperatureResponse xmlns:ns1="http://ufsc.br/previsao">
      <return>13.2</return>
    </ns1:getMinTemperatureResponse>
  </S:Body>
</S:Envelope>
```

**SOAP + attachments**

MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary;
type=text/xml;
start="<soapmsg.xml@example.com>"
--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: soapmsg.xml@example.com

```xml
  <SOAP-ENV:Body>
    <Person>
      <Picture href="http://example.com/myPict.jpg" />
    </Person>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
--MIME_boundary--
```
WSDL - Web Services Description Language

- Language for describing Web services
  - W3C Standard
  - XML based
  - Describes the interface of a Web service
  - Equivalent to Corba IDL description
  - Platform independent description
  - Extensible language
  - A de facto industry standard.

Using WSDL

- Allows tools to generate compatible client and server stubs.
- Allows industries to define standardized service interfaces.
- Allows advertisement of service descriptions, enabling dynamic discovery and binding of compatible services.
- Provides a normalized description of heterogeneous applications.
WSDL Structure

- **portType**
  - Abstract definition of a service (set of operations)
- **Multiple bindings per portType:**
  - How to access it
  - SOAP, JMS, direct call
- **Ports**
  - Where to access it

WSDL elements

- **Types:** type definitions using *XML-Schema*
- **Messages:** describes what goes on the data flows, using the types defined using *XML-Schema*
- **Port types:** collections of related operations, using messages to exchange arguments and results
- **Bindings:** associate port types with protocols (e.g., HTTP GET/POST) and data formats
- **Ports:** associate bindings with network addresses
- **Services:** collection of related ports
Example: Shopping Cart

WSDL definitions

```xml
<definitions name="ShoppingCartDefinitions"
    targetNamespace="http://example.com/ShoppingCart.wsdl"
    xmlns:tns="http://example.com/ShoppingCart.wsdl"
    xmlns:xsd1="http://example.com/ShoppingCart.xsd"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
```

A WSDL document

```xml
<definitions name="ShoppingCartDefinitions"
    xmlns="http://schemas/xmlsoap.org/wsdl"
    targetNamespace="http://example.com/ShoppingCart.wsdl" … >
  <types> … </types>
  <message name="AddItemInput"> … </message>
  <message name="AddItemOutput"> … </message>
  <portType name="ShoppingCart"> … </portType>
  <binding name="CartHTTPXMLBinding" type="tns:ShoppingCart">…
  <binding name="CartSOAPBinding" type="tns:ShoppingCart">…
  <service name="ShoppingCartService">
    <port name="HTTPXMLCart" binding="tns:CartHTTPXMLBinding">
      …
    <port name="SOAPCart" binding="tns:CartSOAPBinding"> …
  </service>
  <import namespace="..." location="...">
</definitions>
```

Types

```xml
<types>
  <schema targetNamespace="http://myservice.net/carttypes"
    xmlns="http://www.w3.org/2000/10/XMLSchema">
    <complexType name="item"><all>
      <element name="description" type="xsd:string"/>
      <element name="quantity" type="xsd:integer"/>
      <element name="price" type="xsd:float"/>
    </all></complexType>
  </schema>
</types>
```
**Messages**

```xml
<message name="AddItemInput">
    <part name="cart-id" type="xsd:string"/>
    <part name="item" type="carttypes:item"/>
    <part name="image" type="xsd:base64Binary"/>
</message>
```

**Port Types**

```xml
<portType name="ShoppingCart">
    <operation name="AddItem">
        <input message="tns:AddItemInput"/>
        <output message="tns:ACK"/>
        <fault name="BadCartID" message="tns:BadCartID"/>
        <fault name="ServiceDown" message="tns:ServiceDown"/>
    </operation>
    <operation name="RemoveItem"/>
    <operation name="ListItems"/>
</portType>
```
**SOAP Binding**

```xml
<binding name="CartHTTPSOAPBinding" type="tns:ShoppingCart">
  <soap:binding style="RPC" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="AddItem">
    <soap:operation soapAction="http://myservice.net/cart/AddItem"/>
    <input>
      <soap:body use="encoded" namespace="http://myservice.net/cart"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output>
      <soap:body use="encoded" namespace="http://myservice.net/cart"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
    <fault name="BadCartID">
      <soap:body use="encoded" namespace="http://myservice.net/cart"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </fault>
    <fault name="ServiceDown">
      <soap:body use="encoded" namespace="http://myservice.net/cart"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </fault>
  </operation>
</binding>
```

**HTTP Binding**

```xml
<binding name="CartHTTPPostBinding" type="tns:ShoppingCart">
  <http:binding verb="POST"/>
  <operation name="AddItem">
    <http:operation location="/AddItem"/>
    <input>
      <mime:content type="application/x-www-form-urlencoded"/>
    </input>
    <output>
      <mime:content type="application/x-www-form-urlencoded"/>
    </output>
    <fault name="BadCartID">
      <mime:mimeXML/>
    </fault>
    <fault name="ServiceDown">
      <mime:content type="application/x-www-form-urlencoded"/>
    </fault>
  </operation>
</binding>
```
Ports

<port name="SOAPCart" binding="tns:SOAPCartBinding">
    <soap:address location="http://myservice.net/soap/cart"/>
</port>

<port name="HTTPPostCart" binding="tns:HTTPPostCartBinding">
    <http:address location="http://myservice.net/cart"/>
</port>

Services

<service name="ShoppingCartService">
    <documentation>A Shopping Cart for the Web</documentation>
    <port name="HTTPPostCart" binding="tns:HTTPPostCartBinding">
        <http:address location="http://myservice.net/cart"/>
    </port>
    <port name="SOAPCart" binding="tns:SOAPCartBinding">
        <soap:address location="http://myservice.net/soap/cart"/>
    </port>
</service>
UDDI

- Defines the operation of a service registry:
  - Data structures for registering
    - Businesses
    - Technical specifications: tModel is a keyed reference to a technical specification.
    - Service and service endpoints: referencing the supported tModels
  - SOAP Access API
  - Rules for the operation of a global registry
    - “private” UDDI nodes are likely to appear, though.

UDDI Basic Structure
References to Taxonomies

- businessEntity
  - XY323eS472384wZ2f3100293
  - Harbour Flowers
  - www.harbourflowersltd.co.au
  - "Serving Inner Sydney Harbour for ..."
  - contacts
  - businessServices
  - identifierBag

- keyedReference
  - EE123
  - NAICS 02417

- keyedReference
  - DFE-2B
  - DUNS 45231

- Peter Smythe
  - 872-6651
  - 4281 King's Blvd, Sydney, NSW
  - Petersmythe22@hotmail.com

- businessService
  - 23701e54683nf...
  - Online catalog
  - "Website where you can ..."
  - BindingTemplates

- BindingTemplate
  - 352041263-44EE...
  - http://www.sydneynet/hamour...
  - tModelInstanceDetails

- tModelInstanceInfo
  - 4453DFC-223C-3ED6...

- tModel References, each with a unique identifier

---

API SOAP para o UDDI

**API de consulta**
- Busca
  - find_business
  - find_service
  - find_binding
  - find_tModel

- Consulta a detalhes
  - get_businessDetail
  - get_serviceDetail
  - get_bindingDetail
  - get_tModelDetail

**API de publicação**
- Adição
  - save_business
  - save_service
  - save_binding
  - save_tModel

- Remoção
  - delete_business
  - delete_service
  - delete_binding
  - delete_tModel

- Segurança
  - get_authToken
  - discard_authToken
Major Challenges in Web Services

- **Discovery**: find available resource in the Web to meet specific needs
- **Selection**: choose the most suitable resources, by several criteria (e.g., cost, matching interfaces)
- **Composition**: design, enact and synchronize (“choreograph”) distributed processes on the Web, using Web services as basic building blocks
Topics

- Introduction
- Web Services (WS)
- Semantic Web (SW)
- Semantic Web Services (SWS)
- Some Major Efforts towards SWS
  - WSDL-S
  - OWL-S
  - SWSF (SWSO + SWSL)
  - WSMO (WSMO + WSML + WSMX)
- Software Tools: WSMT, WSMX, IRS-III, ...
- Case study: Travelling to SBBD

WS standards lack of semantics!

Problem: No way to describe services and data semantics for machine processing in order to support automated service discovery, selection, composition, ...
Deficiencies of WS Technology

- Only syntactical information descriptions and syntactic support for discovery, composition and execution
  
  => *Web Service reuse and integration needs to be done manually*

- No semantic markup for contents / services
  
  => *Current Web Service Technology Stack failed to realize the promise of Web Services*

The Semantic Web

"*The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*"


Means:

- Standards for representing data and metadata
- Semantic descriptions attached to data and services
- Reasoning (e.g., inference) based on semantic descriptions
Ontology in Philosophy

- **From Greek:**
  - Ontos = being, logos = science
- **Shared understanding** of some domain of interest which
  - Conceived as a set of concepts (e.g., entities, attributes, processes), their definitions and inter-relationships.
  - Referred to as a conceptualization.
  - May be used as a unifying framework to solve the above problems in the above described manner.
  - Entails some sort of world view [with respect to a given domain].

Ontologies in Computer Science

- **Shared conceptualizations** of a domain
- **Explicit** and **Formal** to enable its use by machines
- Can have different forms:
  - **Thesaurus** with the semantic relationships between terms (e.g., synonym, holonym (IS_A), meronym (PART_OF))
  - **Taxonomy**
  - **Class Diagram**
  - **Knowledge Base**
    - Classes, properties and their relationships
    - Instances of classes
Ontology Example – Trip

Applications

- **Accurate search of resources**
  Ex.: Student(Mary), Model(Mary), Saint(Mary)

- **Interoperability**

- **Intelligent agents**

- **Reuse and composition of resources**

- **Semantically enabled services & workflows**
Semantic Web Standards (2006)

User Interface & Applications

Trust

Proof

Unifying Logic

Ontology: OWL

Rule: RIF

 SparQL

RDF-S

RDF

XML

URI/IRI

Semantic Heterogeneity in XML

<object id="a1" class="artifact">
    <tuple>
        <title>Nymphes</title>
        <year>1897</year>
        <creator>Monet</creator>
        <price>10,000,000</price>
        <owners refs="p1,p2,p3"/>
    </tuple>
</object>

<object id="p3" class="person">
    <tuple>
        <name>Claudia</name>
        <age>17</age>
    </tuple>
</object>

<work>
    <artist>Monet</artist>
    <name>Nymphes</name>
    <style>Impressionist</style>
    <size>21 x 61</size>
    <cplace>Givern</cplace>
</work>

<work>
    <artist>Monet</artist>
    <title>Waterloo Bridge</title>
    <style>Impressionist</style>
    <size>29.2 x 46.4</size>
    <history>
        Painted with
        <tech>Oil on canvas</tech>
        in...
    </history>
</work>
XML as a Standard Data Model

ODMG Model

| Class: class | Type: V (int v bool v float v string) |
| YAT: &YAT V Any |

| artifacts: set | &Artifact |
| &Artifact |
| tuple |
| &Person |

“Artfact” Schema

“Artworks” Structure

RDF – Resource Description Framework

A standard language & model for expressing semantics on the Semantic Web.

An RDF statement is a triple of the form:

- **Resource**: anything that has a URL
- **Property**: any property of the resource
- **Value**: a literal or another resource

RDF-Schema defines the classes of resources, the possible properties for each class of resource, and the possible values for these properties.

The proposed standard formats for representing ontologies on the Semantic Web (e.g., DAML+OIL, OWL) are extensions of RDF.
RDF’s XML Syntax

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:RDF="http://www.w3.org/RDF/RDF/
xmlns:dc="http://dublincore.org/dc">
  <rdf:Description about="http://www.nbi.cnptia.embrapa.br">
    <dc:name>Núcleo de Bioinformática</dc:name>
    <leader>
      rdf:resource="http://www.cbi.cnptia.embrapa.br/~neshich"
    </leader>
  </rdf:Description>
  <rdf:Description about="http://www.nbi.cnptia.embrapa.br/~neshich">
    <dc:name>Goran Neshich</dc:name>
    <title>Dr.</title>
    <dc:email>neshich@cnptia.embrapa.br</dc:email>
  </rdf:Description>
</rdf:RDF>
```

RDF graph-like structure
Metadata in RDF
Water Balance (same place and institution)

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:RDF="http://www.w3.org/RDF/RDF/"
          xmlns="http://agric.gov.br/DocStd/"
>
  <rdf:Description about="http://www.agric.gov.br/public/WaterBal1234">
    <Source>
      rdf:resource="http://www.cepagri.unicamp.br"
      rdf:resource="http://www.ciagro.iac.gov.sp.br"
    </Source>
    <InitialDate>28/03/2002</InitialDate>
    <FinalDate>31/03/2002</FinalDate>
    <keyword>Water available in Soil</keyword>
    <local>
      rdf:resource="http://www.ibge.gov.br/state_SP"
    </local>
    <measuremnt_unit>
      rdf:resource="http://www.inmetro.gov.br/mm"
    </measuremnt_unit>
  </rdf:Description>
</rdf:RDF>
```
<rdf:Property rdf:about="&AgricZoning;countryOfState"
  a:maxCardinality="1"
  a:minCardinality="1"
  rdfs:label="countryOfState">
  <rdfs:domain rdf:resource="&AgricZoning;State"/>
  <rdfs:range rdf:resource="&AgricZoning;Country"/>
  <a:inverseProperty rdf:resource="&AgricZoning;statesOfCountry"/>
</rdf:Property>

RDF-Schema

<RDF-Schema>

<rdf:Property rdf:about="&AgricZoning;statesOfCountry"
  a:minCardinality="1"
  rdfs:label="statesOfCountry">
  <rdfs:domain rdf:resource="&AgricZoning;Country"/>
  <rdfs:range rdf:resource="&AgricZoning;State"/>
  <a:inverseProperty rdf:resource="&AgricZoning;countryOfState"/>
</rdf:Property>

</RDF-Schema>
One RDF instance

```xml
<AgricZoning:Country rdf:about="&AgricZoning;pais_55" AgricZoning:nameBR="BRASIL" rdfs:label="BRASIL">
  <AgricZoning:officialRegionsOfCountry rdf:resource="&AgricZoning;regof_1"/>
  <AgricZoning:officialRegionsOfCountry rdf:resource="&AgricZoning;regof_2"/>
  <AgricZoning:officialRegionsOfCountry rdf:resource="&AgricZoning;regof_3"/>
  <AgricZoning:officialRegionsOfCountry rdf:resource="&AgricZoning;regof_4"/>
  <AgricZoning:officialRegionsOfCountry rdf:resource="&AgricZoning;regof_5"/>
  <AgricZoning:statesOfCountry rdf:resource="&AgricZoning;estado_11"/>
  <AgricZoning:statesOfCountry rdf:resource="&AgricZoning;estado_12"/>
  <AgricZoning:metroAreasOfCountry rdf:resource="&AgricZoning;metro_5201"/>
</AgricZoning:Country>
```

RDF-Schema to provide Alternative Knowledge Views
**OWL – Ontology Web Language**

- Extends the RDF with standard vocabulary and constructs to define:
  - Local scope of properties
  - Disjointness of classes
  - Boolean combinations of classes
  - Cardinality restrictions
  - Special characteristics of properties (e.g., transitive properties, inverted properties)

**The 3 flavours of OWL**

- **OWL Lite**
  - Restricted expressiveness: excludes enumerated classes, disjointness statements, and arbitrary cardinality
  - Easier to grasp and implement

- **OWL DL**
  - Equivalent to Description Logics
  - Still permits efficient reasoning support
  - Every legal OWL DL document is a legal RDF document (but not the inverse)

- **OWL Full**
  - Fully upward-compatible with RDF, both syntactically and semantically
  - Any valid RDF/RDF Schema conclusion is also a valid OWL Full conclusion
  - Undecidable
Disjunction and Equivalence of Classes

<owl:Class rdf:about="#associateProfessor">
  <owl:disjointWith rdf:resource="#professor"/>
  <owl:disjointWith rdf:resource="#assistantProfessor"/>
</owl:Class>

<owl:Class rdf:ID="faculty">
  <owl:equivalentClass rdf:resource="#academicStaffMember"/>
</owl:Class>

Inverse properties

<owl:ObjectProperty rdf:ID="teaches">
  <rdfs:range rdf:resource="#course"/>
  <rdfs:domain rdf:resource="#academicStaffMember"/>
  <owl:inverseOf rdf:resource="#isTaughtBy"/>
</owl:ObjectProperty>
Abstract OWL Syntax

Class(Person partial
  restriction (hasChild allValuesFrom(Person)))

Class(Parent complete
  Person
  restriction (hasChild someValuesFrom(Person)))

ObjectProperty(hasChild)

Individual (John type(Person)
  value(hasChild Mary))

Rule Languages

Rules:
- father(?x, ?z) v mother(?x, ?z) ⇒ parent(?x, ?z)
- parent(?x, ?z) ∧ parent(?y, ?z) ⇒ brother(?x, ?y)

Knowledge base:
- father(_Fileto, _Claudio)
- father(_Guiga, _Claudio)

Query:
- brother(_Fileto, ?b)
  ⇒ Yes !!
  ⇒ ?b = _Guiga
  ⇒ ?b = _xyz1, _xyz2, _xyz3, _xyz4, ...
SPARQL

W3C recommendation for querying RDF repositories

Exemplo de expressão em SPARQL:

```sparql
SELECT ?concept, ?property, "Mary"
WHERE {
  ?concept property:hasProperty ?property
  FILTER(?property, "name")
}
```

Research Challenges in the Semantic Web

- **Ontology Creation and Management**
  - (Semi)-Automated Ontology Building (+)
  - (Semi)-Automated Semantic Annotation (+)
  - (Semi)-Automated Ontology Evolution (+/-)
  - Reasoning with Inconsistent Ontologies (-)
  - Ontology Mapping, Mediation and Alignment (+)

- **Ontology Use**
  - Ontologies for Knowledge Management (+/-)
  - Semantic IR / Search (++)
  - Semantic Web Services (++ -> future)

- **Tools and Application (++)**
Topics

- Introduction
- Web Services (WS)
- Semantic Web (SW)

**Semantic Web Services (SWS)**
- Some Major Efforts towards SWS
  - WSDL-S
  - OWL-S
  - SWSF (SWSO + SWSL)
  - WSMO (WSMO + WSML + WSMX)
- **Software Tools**: WSMT, WSMX, IRS-III, ...
- **Case study**: Travelling to SBBD

Semantic Web & Web Services
The SWS Vision

- Huge in terms of users and information available
- Ubiquitous

WWW

URI, HTML, HTTP

Deficiencies in Automated Information Processing
- finding
- extraction
- representation
- interpretation
- maintenance

WWW

URI, HTML, HTTP

Semantic Web

RDF, RDF(S), OWL
The SWS Vision

Dynamic

Web Services
UDDI, WSDL, SOAP

Enable Computing over the Web

WWW
URI, HTML, HTTP

Semantic Web
RDF, RDF(S), OWL

Static

Syntactic

Semantic

The SWS Vision

Dynamic

Automated Web services using Bringing the Web to its full potential

Web Services
UDDI, WSDL, SOAP

Semantic Web Services

Semantic Web
RDF, RDF(S), OWL
Semantic description of Web Services

- Should describe all information necessary to enable automated discovery, composition, execution, etc.

- Semantically enhanced repositories

- Tools and platforms that:
  - semantically enrich current Web content
  - facilitate discovery, composition and execution

Semantic Web Services

- define exhaustive description frameworks for describing Web Services and related aspects (Web Service Description Ontologies)

- support ontologies as underlying data model to allow machine supported Web data interpretation (Semantic Web aspect)

- define semantically driven technologies for automation of the Web Service usage process (Web Service aspect)
What (partial) automation should SWS provide?

- **Publication**: Make available the description of the capability of a service
- **Discovery**: Locate different services suitable for a given task
- **Selection**: Choose the most appropriate services among the available ones
- **Composition**: Combine services to achieve a goal
- **Mediation**: Solve mismatches (data, protocol, process) among the combined
- **Execution**: Invoke services following programmatic conventions
- **Monitoring**: Control the execution process
- **Compensation**: Provide transactional support and undo or mitigate unwanted effects
- **Replacement**: Facilitate the substitution of services by equivalent ones

Topics

- **Introduction**
- **Web Services (WS)**
- **Semantic Web (SW)**
- **Semantic Web Services (SWS)**

**Some Major Efforts towards SWS**
- WSDL-S
- OWL-S
- SWSF (SWSO + SWSL)
- WSMO (WSMO + WSML + WSMX)

- **Software Tools**: WSMT, WSMX, IRS-III, ...
- **Case study**: Travelling to SBBD
Some Major SWS Proposals

- **WSDL-S**: extends **WS** technology with semantic descriptions
- **OWL-S**: extends **OWL** for semantically describing **WS**
- **SWSF (SWSO + SWSL)**: roots in **OWL-S** and the **PSL** (Process Specification Language)
- **WSMO (WSMO + WSML + WSMX)**: ontologies, Web services, **goals**, and **mediators**

WSDL-S

- Rather **minimalist and lightweight** approach that extends **WSDL** service descriptions with semantics
- Roots on METEOR-S project, from **Amit Sheth** at LSDIS, Athens, Georgia
- The semantic model is outside WSDL-S, making it impartial to ontology representation language
- Builds upon and stays close to existing industry standards, promoting an upwardly compatible mechanism for adding semantics to Web services
- Support for XML Schema datatype annotations needs to be added to XML-Schema
- Originates of **SAWSDL** (Semantic Annotations for WSDL), W3C's recommendation with IBM
**WSDL-S**

OWL-S is an OWL ontology to describe Web services.

OWL-S leverages on OWL to:
- Support capability based discovery of Web services
- Support automatic composition of Web Services
- Support automatic invocation of Web services

"Complete do not compete"
- OWL-S does not aim to replace the Web services standards rather OWL-S attempts to provide a semantic layer
  - OWL-S relies on WSDL for Web service invocation (see Grounding)
  - OWL-S Expands UDDI for Web service discovery (OWL-S/UDDI mapping)
OWL-S Upper Ontology

- Mapping to WSDL
  - communication protocol (RPC, HTTP, ...)
  - marshalling/serialization
  - transformation to and from XSD to OWL

- Capability specification
- General features of the Service
- Quality of Service
- Classification in Service taxonomies

- Control flow of the service
- Black/Grey/Glass Box view
- Protocol Specification
- Abstract Messages

Service Profiles

- Service Profile
  - Presented by a service.
  - Represents what the service provides
  - Two main uses:
    1. Advertisements of Web Services capabilities (non-functional properties, QoS, Description, classification, etc.)
    2. Request of Web services with a given set of capabilities

- Profile does not specify use/invocation!
OWL-S Service Profile

Capability Description

- **Preconditions**
  - Set of conditions that should hold prior to service invocation

- **Inputs**
  - Set of necessary inputs that the requester should provide to invoke the service

- **Outputs**
  - Results that the requester should expect after interaction with the service provider is completed

- **Effects**
  - Set of statements that should hold true if the service is invoked successfully.

- **Service type**
  - What kind of service is provided (e.g., selling vs. distribution)

- **Product**
  - Product associated with the service (e.g., travel vs. books vs auto parts)

---

Process Model

- **Process Model**
  - Describes how a service works: internal processes of the service
  - Specifies service interaction protocol
  - Specifies abstract messages: ontological type of information transmitted

- **Facilitates**
  - Web service invocation
  - Composition of Web services
  - Monitoring of interaction
**Definition of Process**

A Process represents a transformation (function). It is characterized by four parameters:

- **Inputs**: the inputs that the process requires
- **Preconditions**: the conditions that are required for the process to run correctly
- **Outputs**: the information that results from (and is returned from) the execution of the process
- **Results**: a process may have different outcomes depending on some condition
  - **Condition**: under what condition the result occurs
  - **Constraints on Outputs
  - **Effects**: real world changes resulting from the execution of the process

**Example of an atomic Process**

```xml
<process:AtomicProcess rdf:ID="LogIn">
  <process:hasInput rdf:resource="#AcctName"/>
  <process:hasInput rdf:resource="#Password"/>
  <process:hasOutput rdf:resource="#Ack"/>
  <process:hasPrecondition isMember(AccName)/>
  <process:hasResult>
    <process:inCondition>
      <expr:SWRL-Condition>
        correctLoginInfo(AccName,Password)
      </expr:SWRL-Condition>
    </process:inCondition>
    <process:withOutput rdf:resource="#Ack"/>
    <valueType rdf:resource="#LoginAcceptMsg"/>
  </process:hasResult>
  <process:hasEffect>
    <expr:SWRL-Condition>
      loggedIn(AccName,Password)
    </expr:SWRL-Condition>
    <process:withOutput rdf:resource="#Ack"/>
    <valueType rdf:resource="#LoginAcceptMsg"/>
  </process:hasEffect>
</process:AtomicProcess>```
Ontology of Processes

Process Model Organization

- **Process Model is described as a tree structure**
  - Composite processes are internal nodes
  - Simple and Atomic Processes are the leaves

- **Simple processes represent an abstraction**
  - Placeholders of processes that aren’t specified
  - Or that may be expressed in many different ways

- **Atomic Processes correspond to the basic actions that the Web service performs**
  - Hide the details of how the process is implemented
  - Correspond to WSDL operations

~ related Process Definition Languages a la BPEL
Composite Processes

- Composite Processes specify how processes work together to compute a complex function
- Composite processes define
  1. **Control Flow**
     Specify the temporal relations between the executions of the different sub-processes (sequence, choice, etc.)
  2. **Data Flow**
     Specify how the data produced by one process is transferred to another process

Service Grounding

- Service Grounding
  - Provides a specification of service access information.
  - Service Model + Grounding give everything needed for using the service
  - Builds upon **WSDL** to define message structure and physical binding layer
- Specifies:
  - communication protocols, transport mechanisms, communication languages, etc.
Mapping OWL-S / WSDL 1.1

- **Operations**
  - correspond to Atomic Processes

- **Input/Output**
  - messages correspond to Inputs/Outputs of processes

Result of using the Grounding

- **Invocation mechanism for OWL-S**
  - Invocation based on WSDL
  - Different types of invocation supported by WSDL can be used with OWL-S

- **Clear separation between service description and invocation/implementation**
  - Service description is needed to reason about the service
    - Decide how to use it
    - Decide what information to send and what to expect
  - Service implementation may be based on SOAP and XSD types
  - The crucial point is that the information that travels on the wires and the information used in the ontologies is the same

- **Allows any web service to be represented using OWL-S**
SWSF – Semantic Web Services Framework (SWSO + SWSL)

- Based on OWL-S and PSL (Process Specification Language)
- Richer behavioural process model based on PSL
- Two major components:
  - conceptual model to specify ontologies, called SWSO, and
  - a richer language, called SWSL
- Two variants of SWSL:
  - SWSL-FOL, based on FLOWS (First-order Logic Ontology for Web Services),
  - SWSL-Rules, based on ROWS (Rule Ontology for Web Services)
- Submitted to W3C in 2005
- Standardized by ISO 18269

FLOWS Extensions based on PSL

- Exceptions
- Control constraints
- Occurrence constraints
- Ordering constraints
- State constraints
- FLOWS Core
- PSL Outer Core
WSMO

- WSMO is an ontology and conceptual framework to describe Web services and related aspects
- Based Web Service Modeling Framework (WSMF)
- WSMO is a SDK-Cluster Working Group

The WSMO approach for SWS

- A Conceptual Model for SWS
- A Formal Language for WSMO
- An Execution Environment for WSMO
**WSMO Principles**

- **Web Compliance**
  - XML, URI (IRI), namespaces, but not necessarily RDF/S, OWL, ...

- **Ontology-based & Role Separation**
  - Users exist in different contexts

- **Strict Decoupling & Strong Mediation**
  - Autonomous components with mediators for interoperability

- **Interface vs. Implementation**
  - Distinguish interface (= description) from implementation (=program)
  - WSML

- **Execution Semantics**
  - WSMX

- **Services vs Web Services**
  - A **Web service** is a computational entity which is able to achieve a user’s goal by invocation (e.g., sell books, sell air tickets)
  - A **service** is the actual value provided by this invocation

---

**WSMO model in MOF**

```
               MOF
              /     \
meta-meta-model M3 layer
                /     \                └── WSMO
meta-model     M2 layer
          /     \    └── WSMO Descriptions
model         M1 layer
   /     \       └── Concrete Web Services
information M0 layer
```
WSMO Top Level Concepts

Objectives that a client may have when consulting a Web Service

Provide the formally specified terminology of the information used by all other components

Connectors between components with mediation facilities for handling heterogeneities

Semantic description of Web Services

Non-Functional Properties

- Every WSMO elements is described by properties that contain relevant, non-functional aspects of the item
- used for management and element overall description
- Core Properties:
  - Dublin Core Metadata Element Set plus version (evolution support)
  - W3C-recommendations for description type
- Web Service Specific Properties:
  - quality aspects and other non-functional information of Web Services
  - used for Service Selection
Non-Functional Properties

ontologynp

do#title hasValue "WSML example ontology"
do#subject hasValue "family"
do#description hasValue "fragments of a family ontology to provide WSML examples"
do#contributor hasValue { _"http://homepage.uibk.ac.at/~c703240/foaf.rdf", _"http://homepage.uibk.ac.at/~csaa5569/", _"http://homepage.uibk.ac.at/~c703239/foaf.rdf", _"http://homepage.uibk.ac.at/homepage/~c703319/foaf.rdf" }
do#date hasValue date("2004-11-22")
do#format hasValue "text/plain"
do#language hasValue "en-US"
do#rights hasValue _"http://www.deri.org/privacy.html"
wsml#version hasValue "$Revision: 1.13 $"
end

WSMO Ontologies

Objectives that a client may have when consulting a Web Service

Provide the formally specified terminology of the information used by all other components

Connectors between components with mediation facilities for handling heterogeneities

Semantic description of Web Services
Ontology Example

Ontology class

Class ontology
hasNonFunctionalProperty type nonFunctionalProperty
importsOntology type ontology
usesMediator type ooMediator
hasConcept type concept
hasRelation type relation
hasFunction type function
hasInstance type instance
hasAxiom type axiom
Ontology header

wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight“
namespace {
  _"http://www.inf.ufsc.br/~frank/travel/domainOntology#",
  dc _"http://purl.org/dc/elements/1.1#",
  wsml _"http://www.wsmo.org/wsml/wsml-syntax#" }

ontology _"http://www.inf.ufsc.br/~frank/travel/domainOntology.wsml"
nonFunctionalProperties
  dc#date hasValue _date(2008,10,8)
  dc#format hasValue "text/plain"
  dc#contributor hasValue {"Frank Siqueira", "Adina Sirbu", "Renato Fileto"}
  dc#title hasValue {"SBBD Travel Ontology", "Travel Ontology"}
  dc#language hasValue "en-US"
endNonFunctionalProperties

Concepts and relations

concept Country subConceptOf Region
  name ofType _string
  capital impliesType (0 1) City

concept City subConceptOf Region
  name ofType _string
  country ofType Country

concept BrazilCity subConceptOf City

concept Ticket
  from ofType Region
  to ofType Region
  vehicle ofType Vehicle
## Concepts and relations (cont.)

- `concept Place`
  - `isInCity impliesType (0 1) City`

- `concept Airport subConceptOf Place`

- `concept BusStation subConceptOf Place`

- `concept TrainStation subConceptOf Place`

- `concept PersonsHome subConceptOf Place`

- `concept UniversityCampus subConceptOf Place`

## Instances

- `instance Brazil memberOf Country`
  - `name hasValue "Brazil"`
  - `capital hasValue Brasilia`

- `instance SP memberOf BrazilState`
  - `name hasValue "São Paulo"`
  - `country hasValue Brazil`

- `instance Brasilia memberOf BrazilCity`
  - `name hasValue "Brasília"`
  - `country hasValue Brazil`
**Instances (cont.)**

instance UNICAMP-BaraoGeraldo memberOf UniversityCampus
   isInCity hasValue Campinas

instance UFSC_Trindade memberOf UniversityCampus
   isInCity hasValue Florianopolis

instance HercilioLuz memberOf Airport
   isInCity hasValue Florianopolis

instance Viracopos memberOf Airport
   isInCity hasValue Campinas

instance Congonhas memberOf Airport
   isInCity hasValue SaoPaulo

instance FrancoMontoro memberOf Airport
   isInCity hasValue Guarulhos

**Axioms**

axiom **UKCityDef**
   definedBy
     ?city memberOf UKCity
     implies
     ?city[country hasValue UK]

axiom **BrazilCityDef**
   definedBy
     ?city memberOf BrazilCity
     implies
     ?city[country hasValue Brazil]
**WSMO Goals**

Objectives that a client may have when consulting a Web Service

- **Goals**
- **Ontologies**
- **Mediators**
- **Web Services**

Semantic description of Web Services:
- **Capability** (functional)
- **Interfaces** (usage)

Provide the formally specified terminology of the information used by all other components

Connectors between components with mediation facilities for handling heterogeneities

**Goal class**

Class goal
- hasNonFunctionalProperty type nonFunctionalProperty
- importsOntology type ontology
- usesMediator type {ooMediator, ggMediator}
- requestsCapability type capability multiplicity = single–valued
- requestsInterface type interface
Goals

- **De-coupling of Request and Service**
  - **Goal-driven Approach**, derived from AI rational agent approach
    - Requester formulates objective independent / without regard to services for resolution
    - ‘Intelligent’ mechanisms detect suitable services for solving the Goal
    - Allows re-use of Goals

- **Usage of Goals within Semantic Web Services**
  - A Requester, that is an agent (human or machine), defines a Goal to be resolved
  - Web Service Discovery detects suitable Web Services for solving the Goal automatically
  - Goal Resolution Management is realized in implementations

Goal Example

```xml
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

namespace
  {"_"http://www.inf.ufsc.br/~frank/travel/goalFloripaCampinasSBBD2008#",
  dO _"http://www.inf.ufsc.br/~frank/travel/domainOntology#",
  dc _"http://purl.org/dc/elements/1.1#"

/* Test Goal */
goal
  _"http://www.inf.ufsc.br/~frank/travel/goalFloripaCampinasSBBD2008.wsml"
  nfp
  dc#title hasValue "Goal"
  dc#contributor hasValue "Frank Siqueira, Renato Fileto"
  dc#description hasValue "Buying a ticket from Floripa to Campinas"
endnfp
importsOntology _"http://www.inf.ufsc.br/~frank/travel/domainOntology.wsml"
```
Goal Example (cont)

```
capability goalCapability
postcondition
definedBy
  ?ticket[
    dO#from hasValue ?from,
    dO#to hasValue ?to,
    dO#vehicle hasValue ?vehicle
  ] memberOf dO#Ticket and
  ?from = dO#Florianopolis and
  ?to = dO#Campinas.
```

WSMO Web Services

Objectives that a client may have when consulting a Web Service

- Semantic description of Web Services:
  - Capability (functional)
  - Interfaces (usage)

Provide the formally specified terminology of the information used by all other components

Connectors between components with mediation facilities for handling heterogeneities
WSMO Service

Class service
   hasNonFunctionalProperty type nonFunctionalProperty
   importsOntology type ontology
   usesMediator type \{ooMediator, wwMediator\}
   hasCapability type capability multiplicity = single-valued
   hasInterface type interface

WSMO Web Service

Non-functional Properties -> Capability

Client

Choreography

Web Service Implementation
   (not of interest for Web Service Description)

Orchestration
WSMO Web Service - Interfaces

- complete item description
- quality aspects
- Web Service Management

Non-functional Properties

Core + WS-specific

Capability

functional description

Web Service Implementation
(not of interest in Web Service Description)

Realization of WS by using other Web Services
- Functional decomposition
- WS Composition

Interaction Interface
for consuming WS
- Messages
- External Visible Behavior
- ‘Grounding’

Choreography --- Interfaces --- Orchestration

Web Service specific Properties

- non-functional information of Web Services:
  - Accuracy
  - Availability
  - Financial
  - Network-related QoS
  - Performance
  - Reliability
  - Robustness
  - Scalability
  - Security
  - Transactional
  - Trust
Web Service Example

wsmlVariant "http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"
namespace { _"http://www.inf.ufsc.br/~frank/travel/webServiceBrazilAir#",
  do _"http://www.inf.ufsc.br/~frank/travel/domainOntology#",
  dc _"http://purl.org/dc/elements/1.1#"}

webService _"http://www.inf.ufsc.br/~frank/travel/webServiceBrazilAir.wsml"
  nonFunctionalProperties
    dc#description hasValue "Booking plane tickets within Brazil"
    dc#contributor hasValue "Frank Siqueira"
    dc#title hasValue "Brazil Air"
  endNonFunctionalProperties

importsOntology
  _"http://www.inf.ufsc.br/~frank/travel/domainOntology.wsml"

Web Service Example (cont.)

capability webServiceBrazilAirCapability

postcondition
  definedBy
    ?ticket[
      do#from hasValue ?from,
      do#to hasValue ?to,
      do#vehicle hasValue ?vehicle
    ] memberOf do#Ticket and
    ?from memberOf do#BrazilCity and
    ?to memberOf do#BrazilCity and
    ?vehicle memberOf do#Airplane
Goal-Services Matchmaking

- **Service**: provision of value for some domain
- **Abstract service**: collection of services offered by a provider
- **Goal**: specification of the client needs
  - *E.g.: Booking air tickets from Floripa to Campinas and booking a room in a hotel in Campinas without carpet*
- **Concrete services**: what the provider requires for accessing its services
  - *E.g. Persons’ name, features of the flight, features of the hotel room (maybe a picture)*
- **Web service**: entity using standard interfaces that allow clients to interact with a provider, in order to explore and consume concrete services

---

Heuristic Classification

```
   Abstraction        Matching        Refinement
   |                  |                      |     
   | Abstracted       | Abstracted           |     
   | findings         | diagnoses            |     
   |                   |                      |     
   | Findings          | Diagnosis            |     
   |                   |                      |     
```

---
Services Discovery Process

Possible kinds of matching

- Exact Match
- Plugin Match
- Subsumption Match
- Intersection Match
- Non Match

Goal
Web service
Ontological Coverage
(Fileto et al. 2003)

Consortium(RNA)

Institution(Embrapa) Institution(Unicamp) ...

Unit(CPAC) Unit(CNPTIA) ... Unit(CEPAGRI) ...

Product

... Grain
... Coffee Beans Rice
... Robusta Arabica
... Variety(tupi)

... Grain
... Fruit
... Orange
... Mango
... Pera
... Variety(JAC-2000)

[ Cons(RNA).Inst(Embrapa), Plant.Fruit.Orange, Country(Brazil).Region(SE) ]

Region(SE)

Country(Brazil)

... Region(NE) Region(S) Region(SE)
... State(MG) State(RJ) State(SP) ...

[ Plant.Grain, Country(Brazil).Region(SE) ]

Relating Ontological Coverages for Web Services Discovering

Services providing data about the production of coffee in the Brazilian South-East?

[ Plant, Country(Brazil) ]

[ Plant, Country(Brazil).Region(SE) ]

[ Plant, Country(Brazil).Region(NE) ]

[ Plant, Country(Brazil).State(MG) ]

[ Plant, Country(Brazil).State(SP) ]

[ Plant, Country(Brazil).State(RJ) ]

Web Service 1 Web Service 2 Web Service 3 Web Service 4 Web Service 5 ...
Formal Relationships between Ontological Coverages

Let OC = [ t₁, t₂, tₙ ], OC′ = [ t′₁, t′₂, t′ₙ ] be ontological coverages, where tᵢ, t′ᵢ are terms from the same ontology

- **Overlapping** (reflexive, symmetric, transitive)
  - For all t in OC there exists t’ in OC’ such that
    - t encompasses t’ OR t’ encompasses t
  - For all t’ in OC’ there exists t in OC such that
    - t encompasses t’ OR t’ encompasses t

- **Encompassing** (reflexive, transitive)
  - For all t in OC there exists t’ in OC’ such that t encompasses t’
  - For all t’ in OC’ there exists t in OC such that t encompasses t’

- **Equivalence** (reflexive, symmetric, transitive)
  - For all t in OC there exists t’ in OC’ such that t encompasses t’
  - For all t’ in OC’ there exists t in OC such that t’ encompasses t

WSMO Capabilities/Interfaces

- **Requested/provided:**
  - **Capability** (functional)
  - **Interfaces** (usage)

Objectives that a client may have when consulting a Web Service

Provide the formally specified terminology of the information used by all other components

Connectors between components with mediation facilities addressing heterogeneities

Semantic description of Web Services:
Capability Specification

- **Non functional properties**
- **Imported Ontologies**
- **Used mediators**
  - OO Mediator: importing ontologies as terminology definition
  - WG Mediator: link to a Goal that is solved by the Web Service

- **Pre-conditions**
  What a web service expects in order to be able to provide its service. They define conditions over the input.

- **Assumptions**
  Conditions on the state of the world that has to hold before the Web Service can be executed and work correctly, but not necessarily checked/checkable.

- **Post-conditions**
  describes the result of the Web Service in relation to the input, and conditions on it.

- **Effects**
  Conditions on the state of the world that hold after execution of the Web Service (i.e. changes in the state of the world)

---

Web Service Interfaces

**Choreography**
- request:
  - buyer information, itinerary
  - input not valid
  - no valid connection
  - set of valid itineraries
  - itinerary
  - purchase proposition
  - option selection OR accept OR not accept
  - request payment information
  - payment information incorrect
  - payment information normal
  - successful purchase

**Orchestration**
- connection choice
  - contract of purchase
  - payment & delivery
  - successful purchase

**Internal**
- invocation
  - connection choice
  - contract of purchase
  - payment & delivery

**TimeTable**
- Composition
  - Payment
  - Delivery
Choreography in WSMO

"Interface of Web Service for client-service interaction when consuming the Web Service"

- **External Visible Behavior**
  - those aspects of the workflow of a Web Service where User Interaction is required
  - described by process / workflow constructs

- **Communication Structure**
  - messages sent and received
  - their order (messages are related to activities)

---

Choreography in WSMO (2)

- **Grounding**
  - concrete communication technology for interaction
  - choreography related errors (e.g. input wrong, message timeout, etc.)

- **Formal Model**
  - allow operations / mediation on Choreographies
  - Formal Basis: Abstract State Machines (ASM)

- Very generic description of a transition system over evolving ontologies:
WSMO Orchestration

"Achieve Web Service Functionality by aggregation of other Web Services"

Decomposition of the Web Service functionality into sub functionalities

Proxies: Goals as placeholders for used Web Services

- **Orchestration Language**
  - decomposition of Web Service functionality
  - control structure for aggregation of Web Services

- **Web Service Composition**
  - Combine Web Services into higher-level functionality
  - Resolve mismatches occurring between composed Web Services

- **Proxy Technology**
  - Placeholders for used Web Services or goals, linked via Mediators.
  - Facility for applying the Choreography of used Web Services, service templates for composed services

Choreography & orchestration

**Example:**
WSMO Mediators

Objectives that a client may have when consulting a Web Service

Provide the formally specified terminology of the information used by all other components

Semantic description of Web Services:
- Capability (functional)
- Interfaces (usage)

Mediators

Connectors between components with mediation facilities for handling heterogeneities

Mediation

- Heterogeneity ...
  - Mismatches on structural / semantic / conceptual / level
  - Occur between different components that shall interoperate
  - Especially in distributed & open environments like the Internet

- Concept of Mediation (Wiederhold, 94):
  - Mediators as components that resolve mismatches
  - Declarative Approach:
    - Semantic description of resources
    - 'Intelligent' mechanisms that resolve mismatches independent of content
  - Mediation cannot be fully automated (integration decision)

- Levels of Mediation within Semantic Web Services (WSMF):
  - Data Level: mediate heterogeneous Data Sources
  - Protocol Level: mediate heterogeneous Communication Patterns
  - Process Level: mediate heterogeneous Business Processes
Mediation

Techniques
heterogeneity resolution

Components
connect heterogeneous elements + use mediation techniques

apply

WW Mediator
WG Mediator
GG Mediator
OO Mediator

invocation & execution

Mediation Execution Environment

Mediator Usage

Legend:
- Modelling Element
- Logical connection
Mediators as services

10/11/2008

WSMO Mediator

uses a Mediation Service via

- as a Goal
- directly
- optionally incl. Mediation

Source Component

Source Component

Target Component

Mediation Services

Process Mediation Patterns

(a)
(b)
(c)
(d)
Example of Process Mediation

WSMO Perspective

- WSMO provides a **conceptual model** for Web Services and related aspects
  - WSMO separates the different **language specifications layers** (MOF style)
    - Language for defining WSMO is the meta - meta - model in MOF
    - WSMO and WSML are the meta - models in MOF
    - Actual goals, web services, etc. are the model layer in MOF
    - Actual data described by ontologies and exchanged is the information layer in MOF
  - Stress on solving the integration problem
    - Mediation as a key element
  - Languages to cover wide range of scenarios and improve interoperability
  - Relation to industry WS standards
  - All the way from conceptual modelling to usable implementation (WSML, WSMX)
  - Language: WSML: human readable syntax, XML exchange syntax, RDF/XML exchange syntax under consideration
**WSML**

**Key features:**

- One syntactic framework for a set of layered languages
- Normative, human-readable syntax
- Separation of conceptual and logical modeling
- Semantics based on well-known formalisms
- WWW language
- Frame-based syntax

**WSML vs OWL**

The relation between WSML and OWL+SWRL is still to be completely worked out:

- WSML-Core is a subset of OWL Lite (DL À Datalog)
- WSML-DL is equivalent to OWL DL
- WSML-Flight (refers to "F-Logic" and "Light" ;-) and extends to the LP variant of F-Logic) but for other languages the relation is still unknown.
**WSML Variants**

- WSML-DL (With nonmonotonic extensions)
- WSML-Core
- WSML-Flight
- F-Logic
- DataLog
- Horn
- WSML-Flight (With nonmonotonic negation)
- WSML-Rule
- First-Order Logic (Without NAF)
- WSML-Full
- First-Order Logic (Without EXISTS)

**WSML Layering**

- WSML-Full
- WSML-DL
- WSML-Rule
- WSML-Flight
- WSML-Core
- First-Order Logic
- Logic Programming
- Description Logics
### Relation to Web Services Technology

<table>
<thead>
<tr>
<th></th>
<th>OWL-S</th>
<th>WSMO</th>
<th>Web Services Infrastructure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discovery</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What it does</td>
<td>Profile</td>
<td>Web Services (capability)</td>
<td>UDDI API</td>
</tr>
<tr>
<td><strong>Choreography</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>How is done</td>
<td>Process Model</td>
<td>Orchestration + choreography</td>
<td>BPEL4WS</td>
</tr>
<tr>
<td><strong>Invocation</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>How to invoke</td>
<td>Grounding+ WSDL/SOAP</td>
<td>Grounding</td>
<td>WSDL/SOAP</td>
</tr>
</tbody>
</table>

- OWL-S and WSMO map to UDDI API adding semantic annotation
- OWL-S and WSMO share a default WSDL/SOAP Grounding
- BPEL4WS could be mapped into WSMO orchestration and choreography
- Mapping still unclear at the level of choreography/orchestration
  - In OWL-S, multi-party interaction is obtained through automatic composition and invocation of multiple parties
  - BPEL allows hardcoded representation of many Web services in the same specification
  - Trade-off: OWL-S support substitution of Web services at run time, such substitution is virtually impossible in BPEL.

### Conclusion: How WSMO Addresses WS problems

- **Discovery**
  - Provide formal representation of capabilities and goal
  - Conceptual model for service discovery
  - Different approaches to web service discovery
- **Composition**
  - Provide formal representation of capabilities and choreographies
- **Invocation**
  - Support any type of WS invocation mechanism
  - Clear separation between WS description and implementation
- **Mediation and Interoperation**
  - Mediators as a key conceptual element
  - Mediation mechanism not dictated
  - (Multiple) formal choreographies + mediation enable interoperation
- **Guaranteeing Security and Policies**
  - No explicit policy and security specification yet
  - Proposed solution will interoperate with WS standards
- The solutions are envisioned maintaining a strong relation with existing WS standards
Topics

- Introduction
- Web Services (WS)
- Semantic Web (SW)
- Semantic Web Services (SWS)
- Some Major Efforts towards SWS
  - WSDL-S
  - OWL-S
  - SWSF (SWSO + SWSL)
  - WSMO (WSMO + WSML + WSMX)

**Software Tools:** WSMT, WSMX, IRS-III, ...

- Case study: Travelling to SBBD

Software Tools for SWS

- Design Tools
  - WSMT (Eclipse, WSMO API, WSMO-Studio, WSMT, DOME)

- Execution Environments
  - WSMX

- Reasoners
  - WSML-2 Reasoner
  - IRS-III
WSMT

WSMT: graph based editing, integration of execution environments

WSMOStudio: form based editing, support for choreographies

DOME: tree based editing, support for versioning & distribution

Eclipse Platform  WSMO API
WSMX Architecture
**WSML2REasoner Framework**

- **Reasoning Request**
- **Result (Variable bindings)**

- **WSML Reasoner Interface**
- **Normalization Steps**
- **Datalog Layer**

- **Symbol Map**
- **MINS Facade**
- **KAON2 Facade**
- **...**

- **MINS**
- **KAON2**
- **...**

**Topics**

- **Introduction**
- **Web Services (WS)**
- **Semantic Web (SW)**
- **Semantic Web Services (SWS)**
- **Some Major Efforts towards SWS**
  - WSDL-S
  - OWL-S
  - SWSF (SWSO + SWSL)
  - WSMO (WSMO + WSML + WSMX)
- **Software Tools:** WSMT, WSMX, IRS-III, ...
- **Case study: Travelling to SBBD**
Case Study: Virtual Travel Agency

SBBD Travelling Ontology
Queries

- ?city[country hasValue Brasil]
- ?city memberOf BrazilCity
- ?country[capital hasValue ?capital]
- ?country[capital hasValue ?capital] and ?capital memberOf EUcity
- ?country[capital hasValue ?capital] and ?city[country hasValue ?country] and ?capital != ?city

Query execution
WSMT Perspectives and Navigator
Other Travel Ontologies III

Mediation
Services

Web Service BrazilAir

capability webServiceBrazilAirCapability

postcondition
definedBy

?ticket[
  dO#from hasValue ?from,
  dO#to hasValue ?to,
  dO#vehicle hasValue ?vehicle
] memberOf dO#Ticket and
?from memberOf dO#BrazilCity and
?to memberOf dO#BrazilCity and
?vehicle memberOf dO#Airplane
Goal FLP-CPS

http://www.inf.ufsc.br/~frank/travel/goalFlorianopolisCampinasSEBD2008.w3wl

Goal Florianópolis-Campinas

capability goalCapability
postcondition
definedBy

?ticket[
  dO#from hasValue ?from,
  dO#to hasValue ?to,
  dO#vehicle hasValue ?vehicle
] memberOf dO#Ticket and
?from = dO#Florianopolis and
?to = dO#Campinas.
**Discovered Web Services FLP-CPS**

![WSMX Monitor interface](image)

**Problem!**
Current WSMO version does not properly support inference on instances?

---

**Goal BrazilAir**

```
capability goalCapability
postcondition definedBy
  ?ticket[
    dO#from hasValue ?from,
    dO#to hasValue ?to,
    dO#vehicle hasValue ?vehicle
  ] memberOf dO#Ticket and
  ?from memberOf dO#BrazilCity and
  ?to memberOf dO#BrazilCity
  ?vehicle memberOf dO#Airplane
```
Discovered Web Services BrazilAir

Goal EUAir

capability goal
postcondition
definedBy

?ticket[
  dO#from hasValue ?from,
  dO#to hasValue ?to,
  dO#vehicle hasValue ?vehicle
] memberOf dO#Ticket and
?fro memberOf dO#EUCity and
?to memberOf dO#EUCity
Discovered Web Services EUAir

Conclusions

- SWS research mixes lots of theory and technology
  - Current Web services technology (WSDL, SOAP, UDDI, ...)
  - Semantic Web technology
  - Sophisticated knowledge representation and reasoning
  - Process/workflow technology (orchestration and choreography)

- Some R&D opportunities/challenges in SWS
  - Automated composition of SWS
  - Domain specific issues
  - Software tools for SWS
References – SWS in general


References – SWS major approaches

References - SWS Composition


Thanks all folks!

Questions?
Suggestions?
Comments?
Complaints?