

A motivating introduction to

Semantic Web and Semantic Web Services

Axel Polleres

Universidad Rey Juan Carlos, Madrid
axel@polleres.net

May 25, 2006

Overview

- ▶ The Semantic Web
 - ▶ Idea
 - ▶ “Layer cake”
 - ▶ RDF and OWL
- ▶ Web Services
 - ▶ Components of SOA
 - ▶ SOAP, WSDL, UDDI
- ▶ Towards Semantic Web Services
 - ▶ Aspects
 - ▶ Usage Tasks
- ▶ Approaches
 - ▶ OWL-S
 - ▶ WSMO
 - ▶ SWSF
 - ▶ WSDL-S

The Semantic Web



<http://imdb.com>

<http://badmovies.org>

- ▶ The Semantic Web promises machine-readable **metadata** annotations of websites allowing to combine and query their content, draw additional inferences, just like you'd deal with a huge database.

The Semantic Web



<http://imdb.com>

<http://badmovies.org>

- ▶ The Semantic Web promises machine-readable **metadata** annotations of websites allowing to combine and query their content, draw additional inferences, just like you'd deal with a huge database.
- ▶ E.g., imagine a “Semantic” search engine gathering metadata on movies and ratings, using an agreed vocabulary, I want to ask database like **queries**, such as: *“Search for science fiction movies which are rated as bad?”*

The Semantic Web



<http://imdb.com>

<http://badmovies.org>

- ▶ The Semantic Web promises machine-readable **metadata** annotations of websites allowing to combine and query their content, draw additional inferences, just like you'd deal with a huge database.
- ▶ E.g., imagine a "Semantic" search engine gathering metadata on movies and ratings, using an agreed vocabulary, I want to ask database like **queries**, such as: *"Search for science fiction movies which are rated as bad?"*
- ▶ I want to express **taxonomies** such as *"Science-fiction movies are movies."*

The Semantic Web



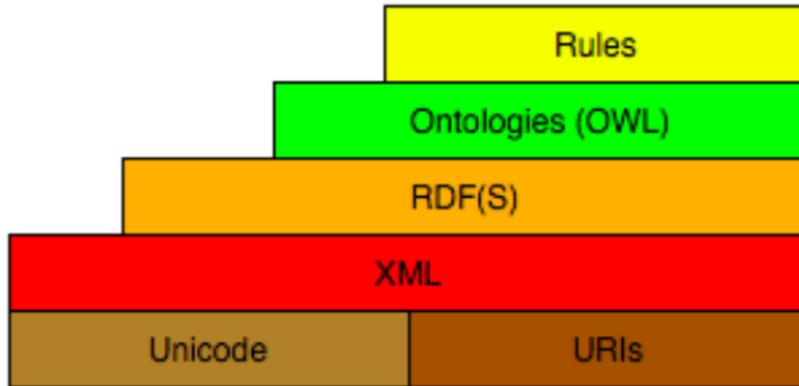
<http://imdb.com>

<http://badmovies.org>

- ▶ The Semantic Web promises machine-readable **metadata** annotations of websites allowing to combine and query their content, draw additional inferences, just like you'd deal with a huge database.
- ▶ E.g., imagine a "Semantic" search engine gathering metadata on movies and ratings, using an agreed vocabulary, I want to ask database like **queries**, such as: *"Search for science fiction movies which are rated as bad?"*
- ▶ I want to express **taxonomies** such as *"Science-fiction movies are movies."*
- ▶ Besides metadata **facts**, I want to express more complex **rules** such as for instance: *"All movies listed on badmovies.org are rated bad."*



The W3C's Semantic Web “layer cake”



- ▶ XML is the basis
- ▶ RDF is a graph-based datamodel for describing meta-data
- ▶ OWL and Rules shall provide possibility to infer additional knowledge

Remark: Semantic Web is not only about combining Web meta-data, but about **data integration** in general (not a new issue)!

RDF in a nutshell...

The RDF is the **data model** for the Semantic Web metadata.

RDF describes a **labeled** graph of resources (nodes) linked to other resources or literals by predicates.



RDF in a nutshell...

The RDF is the **data model** for the Semantic Web metadata.

RDF describes a **labeled** graph of resources (nodes) linked to other resources or literals by predicates.

- ▶ usually represented in form of triples $\langle \textit{Subject}, \textit{Predicate}, \textit{Object} \rangle$ e.g.

RDF in a nutshell...

The RDF is the **data model** for the Semantic Web metadata.

RDF describes a **labeled** graph of resources (nodes) linked to other resources or literals by predicates.

- ▶ usually represented in form of triples $\langle \textit{Subject}, \textit{Predicate}, \textit{Object} \rangle$ e.g.

```
http://www.polleres.net/index.html dc:creator http://www.polleres.net/foaf.rdf#me.  
http://www.polleres.net/foaf.rdf#me foaf:name "Axel Polleres"
```

```
<rdf:Description rdf:about="http://www.polleres.net/index.html">  
  <dc:creator>  
    <rdf:Description rdf:about="http://www.polleres.net/foaf.rdf#me">  
      <foaf:Name>Axel Polleres</foaf:Name>  
    </rdf:Description>  
  </dc:creator>  
</rdf:Description>
```

RDF in a nutshell...

The RDF is the **data model** for the Semantic Web metadata.

RDF describes a **labeled** graph of resources (nodes) linked to other resources or literals by predicates.

- ▶ usually represented in form of triples $\langle \textit{Subject}, \textit{Predicate}, \textit{Object} \rangle$ e.g.

```
http://www.polleres.net/index.html dc:creator http://www.polleres.net/foaf.rdf#me.  
http://www.polleres.net/foaf.rdf#me foaf:name "Axel Polleres"
```

```
<rdf:Description rdf:about="http://www.polleres.net/index.html">  
  <dc:creator>  
    <rdf:Description rdf:about="http://www.polleres.net/foaf.rdf#me">  
      <foaf:Name>Axel Polleres</foaf:Name>  
    </rdf:Description>  
  </dc:creator>  
</rdf:Description>
```

- ▶ Resources identified by URIs, not necessarily only Web pages.

RDF in a nutshell...

The RDF is the **data model** for the Semantic Web metadata.

RDF describes a **labeled** graph of resources (nodes) linked to other resources or literals by predicates.

- ▶ usually represented in form of triples $\langle \textit{Subject}, \textit{Predicate}, \textit{Object} \rangle$ e.g.

```
http://www.polleres.net/index.html dc:creator http://www.polleres.net/foaf.rdf#me.  
http://www.polleres.net/foaf.rdf#me foaf:name "Axel Polleres"
```

```
<rdf:Description rdf:about="http://www.polleres.net/index.html">  
  <dc:creator>  
    <rdf:Description rdf:about="http://www.polleres.net/foaf.rdf#me">  
      <foaf:Name>Axel Polleres</foaf:Name>  
    </rdf:Description>  
  </dc:creator>  
</rdf:Description>
```

- ▶ Resources identified by URIs, not necessarily only Web pages.
- ▶ RDFS allows to define simple taxonomies on RDF vocabularies using `rdf:type`, `rdf:subClassOf`, `rdfs:subPropertyOf`

RDF in a nutshell...

The RDF is the **data model** for the Semantic Web metadata.

RDF describes a **labeled** graph of resources (nodes) linked to other resources or literals by predicates.

- ▶ usually represented in form of triples $\langle \textit{Subject}, \textit{Predicate}, \textit{Object} \rangle$ e.g.

```
http://www.polleres.net/index.html dc:creator http://www.polleres.net/foaf.rdf#me.  
http://www.polleres.net/foaf.rdf#me foaf:name "Axel Polleres"
```

```
<rdf:Description rdf:about="http://www.polleres.net/index.html">  
  <dc:creator>  
    <rdf:Description rdf:about="http://www.polleres.net/foaf.rdf#me">  
      <foaf:Name>Axel Polleres</foaf:Name>  
    </rdf:Description>  
  </dc:creator>  
</rdf:Description>
```

- ▶ Resources identified by URIs, not necessarily only Web pages.
- ▶ RDFS allows to define simple taxonomies on RDF vocabularies using `rdf:type`, `rdf:subClassOf`, `rdfs:subPropertyOf`
- ▶ Some subtleties in RDF semantics (blank nodes, XML literals, RDF keywords treated as normal resources, reification, etc.)

RDF in a nutshell...

The RDF is the **data model** for the Semantic Web metadata.

RDF describes a **labeled** graph of resources (nodes) linked to other resources or literals by predicates.

- ▶ usually represented in form of triples $\langle \textit{Subject}, \textit{Predicate}, \textit{Object} \rangle$ e.g.

```
http://www.polleres.net/index.html dc:creator http://www.polleres.net/foaf.rdf#me.  
http://www.polleres.net/foaf.rdf#me foaf:name "Axel Polleres"
```

```
<rdf:Description rdf:about="http://www.polleres.net/index.html">  
  <dc:creator>  
    <rdf:Description rdf:about="http://www.polleres.net/foaf.rdf#me">  
      <foaf:Name>Axel Polleres</foaf:Name>  
    </rdf:Description>  
  </dc:creator>  
</rdf:Description>
```

- ▶ Resources identified by URIs, not necessarily only Web pages.
- ▶ RDFS allows to define simple taxonomies on RDF vocabularies using `rdf:type`, `rdf:subClassOf`, `rdfs:subPropertyOf`
- ▶ Some subtleties in RDF semantics (blank nodes, XML literals, RDF keywords treated as normal resources, reification, etc.)
- ▶ RDFS can be seen as logic!

RDF in a nutshell...

The RDF is the **data model** for the Semantic Web metadata.

RDF describes a **labeled** graph of resources (nodes) linked to other resources or literals by predicates.

- ▶ usually represented in form of triples $\langle \textit{Subject}, \textit{Predicate}, \textit{Object} \rangle$ e.g.

```
http://www.polleres.net/index.html dc:creator http://www.polleres.net/foaf.rdf#me.  
http://www.polleres.net/foaf.rdf#me foaf:name "Axel Polleres"
```

```
<rdf:Description rdf:about="http://www.polleres.net/index.html">  
  <dc:creator>  
    <rdf:Description rdf:about="http://www.polleres.net/foaf.rdf#me">  
      <foaf:Name>Axel Polleres</foaf:Name>  
    </rdf:Description>  
  </dc:creator>  
</rdf:Description>
```

- ▶ Resources identified by URIs, not necessarily only Web pages.
- ▶ RDFS allows to define simple taxonomies on RDF vocabularies using `rdf:type`, `rdf:subClassOf`, `rdfs:subPropertyOf`
- ▶ Some subtleties in RDF semantics (blank nodes, XML literals, RDF keywords treated as normal resources, reification, etc.)
- ▶ RDFS can be seen as logic! (Don't panic!)

OWL offers more expressivity than RDF/S

Web **Ontology** Language.

- ▶ Ontologies allow to to *formally define shared conceptualizations*, on top of the RDF/RDFS data model.

OWL offers more expressivity than RDF/S

Web **Ontology** Language.

- ▶ Ontologies allow to to *formally define shared conceptualizations*, on top of the RDF/RDFS data model.
- ▶ OWL is basically a variant of description logic $\mathcal{SHOIN}(D)$, enough that we know that it allows to describe **classes** of resources, and their **properties** as well as some restrictions on their use.

OWL offers more expressivity than RDF/S

Web **Ontology** Language.

- ▶ Ontologies allow to to *formally define shared conceptualizations*, on top of the RDF/RDFS data model.
- ▶ OWL is basically a variant of description logic $\mathcal{SHOIN}(D)$, enough that we know that it allows to describe **classes** of resources, and their **properties** as well as some restrictions on their use.
- ▶ Simply: Additional rules, descriptions of a data model in a formal language, related: UML, EER, etc.

OWL offers more expressivity than RDF/S

Web **Ontology** Language.

- ▶ Ontologies allow to to *formally define shared conceptualizations*, on top of the RDF/RDFS data model.
- ▶ OWL is basically a variant of description logic $\mathcal{SHOIN}(D)$, enough that we know that it allows to describe **classes** of resources, and their **properties** as well as some restrictions on their use.
- ▶ Simply: Additional rules, descriptions of a data model in a formal language, related: UML, EER, etc.
- ▶ What makes ontologies different from datamodels is :
Consensual!
- ▶ OWL/RDF are only a languages for this, i.e. Ontologies and the semantic Web only work if people **share** ontologies.

Semantic Web – What for?

- ▶ Well, why is XML/XML Schema not enough? If every body uses the same XML schema, then all is fine, right?

Semantic Web – What for?

- ▶ Well, why is XML/XML Schema not enough? If every body uses the same XML schema, then all is fine, right?
- ▶ But: RDF “flat” data model, is easier to merge/combine than XML trees.

Semantic Web – What for?

- ▶ Well, why is XML/XML Schema not enough? If every body uses the same XML schema, then all is fine, right?
- ▶ But: RDF “flat” data model, is easier to merge/combine than XML trees.
- ▶ OWL offers more expressivity, enables automatic inference

Semantic Web – What for?

- ▶ Well, why is XML/XML Schema not enough? If every body uses the same XML schema, then all is fine, right?
- ▶ But: RDF “flat” data model, is easier to merge/combine than XML trees.
- ▶ OWL offers more expressivity, enables automatic inference
- ▶ Still, only at the start so far, but first RDF/OWL vocabularies gain momentum: e.g. **RSS**, **FOAF**, **Dublin Core**, also **iCAL** has an RDF format which is widely used, etc.

Semantic Web – What for?

- ▶ Well, why is XML/XML Schema not enough? If every body uses the same XML schema, then all is fine, right?
- ▶ But: RDF “flat” data model, is easier to merge/combine than XML trees.
- ▶ OWL offers more expressivity, enables automatic inference
- ▶ Still, only at the start so far, but first RDF/OWL vocabularies gain momentum: e.g. **RSS, FOAF, Dublin Core**, also **iCAL** has an RDF format which is widely used, etc.
- ▶ Take home message: OWL/RDF make it easy to
 - ▶ combine these vocabularies and develop intelligent methods on top.
 - ▶ axiomatize meaning = semantics!

Semantic Web – What for?

- ▶ Well, why is XML/XML Schema not enough? If every body uses the same XML schema, then all is fine, right?
- ▶ But: RDF “flat” data model, is easier to merge/combine than XML trees.
- ▶ OWL offers more expressivity, enables automatic inference
- ▶ Still, only at the start so far, but first RDF/OWL vocabularies gain momentum: e.g. **RSS, FOAF, Dublin Core**, also **iCAL** has an RDF format which is widely used, etc.
- ▶ Take home message: OWL/RDF make it easy to
 - ▶ combine these vocabularies and develop intelligent methods on top.
 - ▶ axiomatize meaning = semantics!
- ▶ Helps us to automatize aggregation and integration of data (**not only**) on the Web!

Semantic Web – What for?

- ▶ Well, why is XML/XML Schema not enough? If every body uses the same XML schema, then all is fine, right?
- ▶ But: RDF “flat” data model, is easier to merge/combine than XML trees.
- ▶ OWL offers more expressivity, enables automatic inference
- ▶ Still, only at the start so far, but first RDF/OWL vocabularies gain momentum: e.g. **RSS, FOAF, Dublin Core**, also **iCAL** has an RDF format which is widely used, etc.
- ▶ Take home message: OWL/RDF make it easy to
 - ▶ combine these vocabularies and develop intelligent methods on top.
 - ▶ axiomatize meaning = semantics!
- ▶ Helps us to automatize aggregation and integration of data (**not only**) on the Web!
- ▶ Again: automatd reasoning and a bit of logic as the foundations!

From static to dynamic



http://www.renfe.es



http://amazon.com

Current Web pages offer not only static data but also **dynamic services**, e.g. buying books, booking hotels, buying train tickets, etc.

- ▶ Question: Can we automatize service usage in a similar way as aggregation/querying of static data?

From static to dynamic



http://www.renfe.es



http://amazon.com

Current Web pages offer not only static data but also **dynamic services**, e.g. buying books, booking hotels, buying train tickets, etc.

- ▶ Question: Can we automatize service usage in a similar way as aggregation/querying of static data?
- ▶ Just like data integration, making applications and software components interoperable/combinable is not a new issue in the IT landscape...

From static to dynamic



http://www.renfe.es



http://amazon.com

Current Web pages offer not only static data but also **dynamic services**, e.g. buying books, booking hotels, buying train tickets, etc.

- ▶ Question: Can we automatize service usage in a similar way as aggregation/querying of static data?
- ▶ Just like data integration, making applications and software components interoperable/combinable is not a new issue in the IT landscape... keyword: **“Middleware”!**

Web Services (1/2)

- ▶ Current “buzzwords” on middleware are: **Service-Oriented Architectures (SOA)**, **Web Services**.
- ▶ Web services denote a set of standards to enable distributed application development based on Web standards.

Web Services (1/2)

- ▶ Current “buzzwords” on middleware are: **Service-Oriented Architectures (SOA)**, **Web Services**.
- ▶ Web services denote a set of standards to enable distributed application development based on Web standards.
- ▶ Four main “ingredients”:
 - ▶ An agreed *transport protocol* (SOAP over HTTP)
 - ▶ An agreed *message description format* (XML Schema, SOAP)
 - ▶ A language for *interface description* (WSDL)
 - ▶ A *registry* for publication and discovery of available services (UDDI)

Web Services (1/2)

- ▶ Current “buzzwords” on middleware are: **Service-Oriented Architectures (SOA)**, **Web Services**.
- ▶ Web services denote a set of standards to enable distributed application development based on Web standards.
- ▶ Four main “ingredients”:
 - ▶ An agreed *transport protocol* (SOAP over HTTP)
 - ▶ An agreed *message description format* (XML Schema, SOAP)
 - ▶ A language for *interface description* (WSDL)
 - ▶ A *registry* for publication and discovery of available services (UDDI)
- ▶ What is “webbish” about Web services?
 - ▶ Using Web protocols such as HTTP, allow easy integration with exiting Web server technologies as “application servers”
 - ▶ Strictly relying on XML as message exchange format

Web Services (2/2)

What makes Web services **different** from its predecessors (CORBA, RMI, DCOM,etc.)?

Web Services (2/2)

What makes Web services **different** from its predecessors (CORBA, RMI, DCOM,etc.)?

- ▶ light-weight

Web Services (2/2)

What makes Web services **different** from its predecessors (CORBA, RMI, DCOM,etc.)?

- ▶ light-weight
- ▶ modular

Web Services (2/2)

What makes Web services **different** from its predecessors (CORBA, RMI, DCOM, etc.)?

- ▶ light-weight
- ▶ modular
- ▶ extensible (e.g. by Semantic Web technology), rely on open standards

Web Services (2/2)

What makes Web services **different** from its predecessors (CORBA, RMI, DCOM, etc.)?

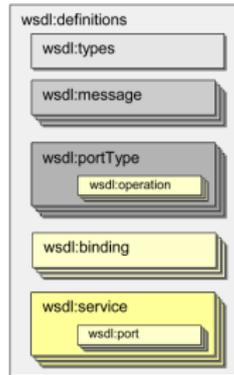
- ▶ light-weight
- ▶ modular
- ▶ extensible (e.g. by Semantic Web technology), rely on open standards
- ▶ Standardization bodies support it: W3C, OASIS (Organization for the Advancement of Structured Information Standards)
- ▶ “Global Players” (IBM, Microsoft, BEA, etc.) collaborate!

⇒ High potential!

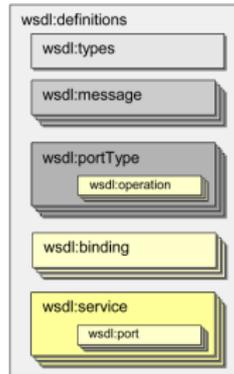
Web Services - SOAP

- ▶ Messaging framework for peers communicating XML messages.
- ▶ packs an XML message in a so-called SOAP “envelope” which can contain additional fault handling and routing information, etc.
- ▶ Most common protocol binding is on top of HTTP, but also other possible.

Web Services - WSDL (1/2)

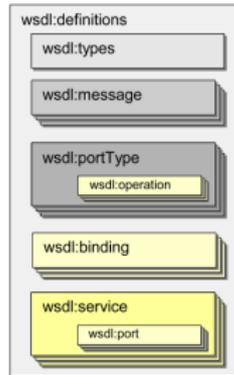


Web Services - WSDL (1/2)



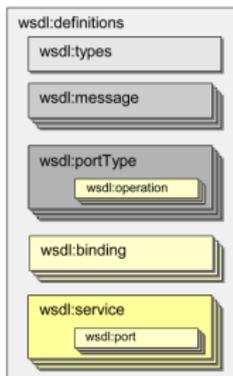
- ▶ Define message types by XML Schema

Web Services - WSDL (1/2)



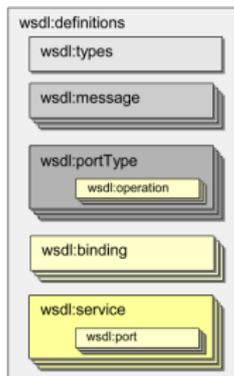
- ▶ Define message types by XML Schema
- ▶ Define messages and operations

Web Services - WSDL (1/2)



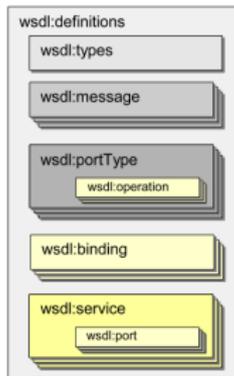
- ▶ Define message types by XML Schema
- ▶ Define messages and operations
- ▶ Group several operations in “ports”

Web Services - WSDL (1/2)



- ▶ Define message types by XML Schema
- ▶ Define messages and operations
- ▶ Group several operations in “ports”
- ▶ Define binding protocol (e.g. SOAP over HTTP, HTTP/GET, etc.)

Web Services - WSDL (1/2)



- ▶ Define message types by XML Schema
- ▶ Define messages and operations
- ▶ Group several operations in “ports”
- ▶ Define binding protocol (e.g. SOAP over HTTP, HTTP/GET, etc.)
- ▶ Define the service *endpoint* address where the service can be invoked.

Web Services - WSDL (2/2)

If you wanna play around, see e.g.: <http://www.xmethods.net/>

```
[...]  
<wsdl:types>  
  [...]  
  <s:element name="GetWeather">  
    <s:complexType>  
      <s:sequence>  
        <s:element minOccurs="0" maxOccurs="1" name="CityName" type="s:string" />  
        <s:element minOccurs="0" maxOccurs="1" name="CountryName" type="s:string" />  
      </s:sequence>  
    </s:complexType>  
  </s:element>  
  [...]  
</wsdl:types>  
  
<wsdl:message name="GetWeatherIn">  
  <wsdl:part name="parameters" element="tns:GetWeather" />  
</wsdl:message>  
[...]  
  
<wsdl:portType name="GlobalWeather">  
  <wsdl:operation name="GetWeather">  
    <wsdl:input message="tns:GetWeatherSoapIn" />  
    <wsdl:output message="tns:GetWeatherSoapOut" />  
  </wsdl:operation>  
  [...]  
</wsdl:portType>  
  
<wsdl:binding name="GlobalWeatherSoap" type="tns:GlobalWeather">  
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />  
  <wsdl:operation name="GetWeather">  
    <soap:operation soapAction="http://www.webservicex.NET/GetWeather" style="document" />  
    <wsdl:input><soap:body use="literal" /></wsdl:input>  
    <wsdl:output><soap:body use="literal" /></wsdl:output>  
  </wsdl:operation>  
</wsdl:binding>
```

Web Services - UDDI

UDDI is

- ▶ an API for publishing and searching Business partners and service providers.
- ▶ a data model for service and business entities
- ▶ allows to link to service classifications (e.g. UNSPC) and technical information (e.g. provided Web services)

Web Services - UDDI

UDDI is

- ▶ an API for publishing and searching Business partners and service providers.
- ▶ a data model for service and business entities
- ▶ allows to link to service classifications (e.g. UNSPC) and technical information (e.g. provided Web services)
- ▶ However: general classifications or keywords in natural language description are insufficient for automatic discovery

Web Services - UDDI

UDDI is

- ▶ an API for publishing and searching Business partners and service providers.
- ▶ a data model for service and business entities
- ▶ allows to link to service classifications (e.g. UNSPC) and technical information (e.g. provided Web services)
- ▶ However: general classifications or keywords in natural language description are insufficient for automatic discovery

Summary: WSDL, SOAP, UDDI operate on a largely “syntactic” level. . . not aligned with Semantic Web standards OWL/RDF, etc.

Web Services - UDDI

UDDI is

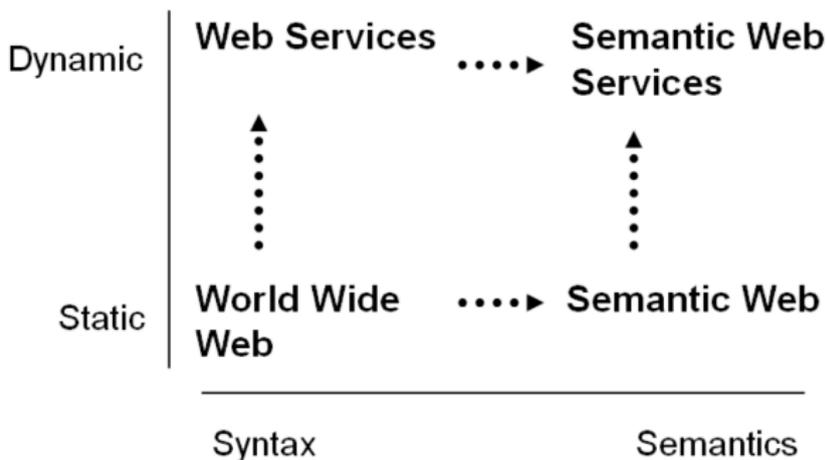
- ▶ an API for publishing and searching Business partners and service providers.
- ▶ a data model for service and business entities
- ▶ allows to link to service classifications (e.g. UNSPC) and technical information (e.g. provided Web services)
- ▶ However: general classifications or keywords in natural language description are insufficient for automatic discovery

Summary: WSDL, SOAP, UDDI operate on a largely “syntactic” level. . . not aligned with Semantic Web standards OWL/RDF, etc.

Would make sense to use the similar metadata format, for annotating services, WSDL operations, input/output messages, etc. to describe their meaning.

What's missing with Web Services?

By combination of Web services with Semantic Web technologies, we hope to achieve a higher degree of automatization of discovery, composition, invocation, etc.



Aim

- ▶ Semantically enhanced repositories

Aim

- ▶ Semantically enhanced repositories
- ▶ Tools and platforms that semantically enrich current Web service descriptions and facilitate:
 - ▶ **Discovery**: Locate different services suitable for a given task

Aim

- ▶ Semantically enhanced repositories
- ▶ Tools and platforms that semantically enrich current Web service descriptions and facilitate:
 - ▶ **Discovery**: Locate different services suitable for a given task
 - ▶ **Selection**: Choose the most appropriate services among these

Aim

- ▶ Semantically enhanced repositories
- ▶ Tools and platforms that semantically enrich current Web service descriptions and facilitate:
 - ▶ **Discovery**: Locate different services suitable for a given task
 - ▶ **Selection**: Choose the most appropriate services among these
 - ▶ **Composition**: Combine services to achieve a goal

Aim

- ▶ Semantically enhanced repositories
- ▶ Tools and platforms that semantically enrich current Web service descriptions and facilitate:
 - ▶ **Discovery**: Locate different services suitable for a given task
 - ▶ **Selection**: Choose the most appropriate services among these
 - ▶ **Composition**: Combine services to achieve a goal
 - ▶ **Mediation**: Solve mismatches (data, protocol, process)

Aim

- ▶ Semantically enhanced repositories
- ▶ Tools and platforms that semantically enrich current Web service descriptions and facilitate:
 - ▶ **Discovery:** Locate different services suitable for a given task
 - ▶ **Selection:** Choose the most appropriate services among these
 - ▶ **Composition:** Combine services to achieve a goal
 - ▶ **Mediation:** Solve mismatches (data, protocol, process)
 - ▶ **Execution:** Invoke services following grammatic conventions

Aim

- ▶ Semantically enhanced repositories
- ▶ Tools and platforms that semantically enrich current Web service descriptions and facilitate:
 - ▶ **Discovery:** Locate different services suitable for a given task
 - ▶ **Selection:** Choose the most appropriate services among these
 - ▶ **Composition:** Combine services to achieve a goal
 - ▶ **Mediation:** Solve mismatches (data, protocol, process)
 - ▶ **Execution:** Invoke services following programmatic conventions
 - ▶ **Monitoring:** Control the execution process

Aim

- ▶ Semantically enhanced repositories
- ▶ Tools and platforms that semantically enrich current Web service descriptions and facilitate:
 - ▶ **Discovery:** Locate different services suitable for a given task
 - ▶ **Selection:** Choose the most appropriate services among these
 - ▶ **Composition:** Combine services to achieve a goal
 - ▶ **Mediation:** Solve mismatches (data, protocol, process)
 - ▶ **Execution:** Invoke services following programmatic conventions
 - ▶ **Monitoring:** Control the execution process
 - ▶ **Compensation:** Transactions, undo/mitigate unwanted effects

Aim

- ▶ Semantically enhanced repositories
- ▶ Tools and platforms that semantically enrich current Web service descriptions and facilitate:
 - ▶ **Discovery:** Locate different services suitable for a given task
 - ▶ **Selection:** Choose the most appropriate services among these
 - ▶ **Composition:** Combine services to achieve a goal
 - ▶ **Mediation:** Solve mismatches (data, protocol, process)
 - ▶ **Execution:** Invoke services following programmatic conventions
 - ▶ **Monitoring:** Control the execution process
 - ▶ **Compensation:** Transactions, undo/mitigate unwanted effects
 - ▶ **Replacement:** Facilitate substitution of services by equivalent ones

Representational Aspects of Semantic service description

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

1. General *service classifications* using taxonomies
2. *pre- and postconditions*, functional aspects (What does the service provide under which conditions?)
3. *behavior/protocol* description of the service (How to interact with the service in order to achieve a certain functionality?)
4. non-functional aspects (QoS, cost, availability, etc.)

Representational Aspects of Semantic service description

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

1. General *service classifications* using taxonomies
2. *pre- and postconditions*, functional aspects (What does the service provide under which conditions?)
3. *behavior/protocol* description of the service (How to interact with the service in order to achieve a certain functionality?)
4. non-functional aspects (QoS, cost, availability, etc.)

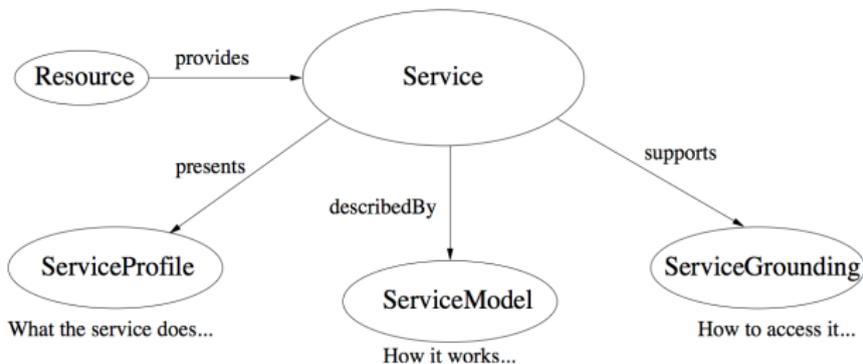
Approaches:

- ▶ OWL-S
- ▶ WSMO
- ▶ SWSF
- ▶ WSDL-S

OWL-S

<http://www.w3.org/Submission/OWL-S/>

- ▶ OWL-S is an OWL ontology to describe Web services, i.e. a metadata vocabulary for services
- ▶ Main components of a service described in three sub-ontologies:



OWL-S Service Profile

Two main uses:

- ▶ Advertisements of Web Services capabilities (non-functional properties, QoS, Description, classification, etc.)
- ▶ Request of Web services with a given set of capabilities

OWL-S Service Profile

Two main uses:

- ▶ Advertisements of Web Services capabilities (non-functional properties, QoS, Description, classification, etc.)
- ▶ Request of Web services with a given set of capabilities

Classes/Properties:

- 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 (eg selling vs distribution)
- Product** Product associated with the service (eg travel vs books vs auto parts)

OWL-S Service Profile

Two main uses:

- ▶ Advertisements of Web Services capabilities (non-functional properties, QoS, Description, classification, etc.)
- ▶ Request of Web services with a given set of capabilities

Classes/Properties:

- 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 (eg selling vs distribution)
- Product** Product associated with the service (eg travel vs books vs auto parts)

Logics: *outside OWL!* Reference to Preconditions/Effects can refer to KIF, DRS, SWRL

OWL-S Service model

Main uses:

- ▶ Define Process Model: Describes how a service works. Internal processes of the service Specifies service, interaction protocol
- ▶ Specify abstract messages (can be inherited or refined from profile): ontological type of information transmitted
- ▶ Facilitate Web service invocation, Composition of Web services Monitoring of interaction

OWL-S Service model

Main uses:

- ▶ Define Process Model: Describes how a service works. Internal processes of the service Specifies service, interaction protocol
- ▶ Specify abstract messages (can be inherited or refined from profile): ontological type of information transmitted
- ▶ Facilitate Web service invocation, Composition of Web services Monitoring of interaction

Classes/Properties:

- ▶ Each process model is built from atomic and composite processes
- ▶ **Atomic processes:**
 - 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. Result consists of: **Condition, Constraints**, real world **Effects**.
- ▶ **Composite processes:** OWL-S defines a simple treelike “workflow language” for defining processes consisting of sequence, loop, switch, parallel execution, etc. (**control flow**) and **dataflow** etc.

OWL-S Service model

Main uses:

- ▶ Define Process Model: Describes how a service works. Internal processes of the service Specifies service, interaction protocol
- ▶ Specify abstract messages (can be inherited or refined from profile): ontological type of information transmitted
- ▶ Facilitate Web service invocation, Composition of Web services Monitoring of interaction

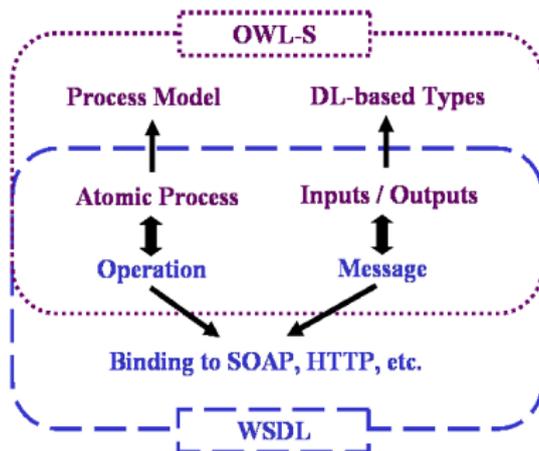
Classes/Properties:

- ▶ Each process model is built from atomic and composite processes
- ▶ **Atomic processes:**
 - 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. Result consists of: **Condition, Constraints**, real world **Effects**.
- ▶ **Composite processes:** OWL-S defines a simple treelike “workflow language” for defining processes consisting of sequence, loop, switch, parallel execution, etc. (**control flow**) and **dataflow** etc.

Problem: OWL (DL) doesn't capture semantics of workflow, conditions, etc.

OWL-S Grounding

Shall close the GAP to “traditional” Web Services world, allow linking to arbitrary WSDL descriptions.



Possible problem: Simple mapping would still allow syntactic differences.

Solution: Last version of OWL-S allows to e.g. link to XSLT to link between ontological representation and XSD defined messages in WSDL.

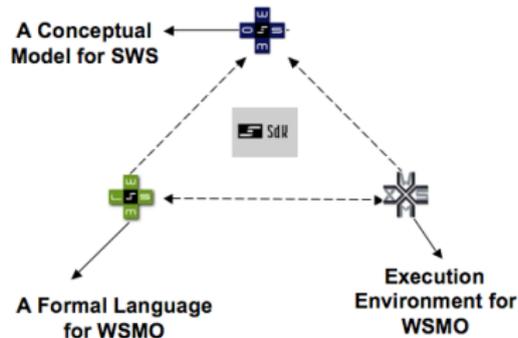
WSMO

<http://www.w3.org/Submission/WSMO/>

European Effort, concept based in PSMs, UMPL, etc. More a framework for SWS annotation than an ontology

Tries to solve some of the OWL-S problems:

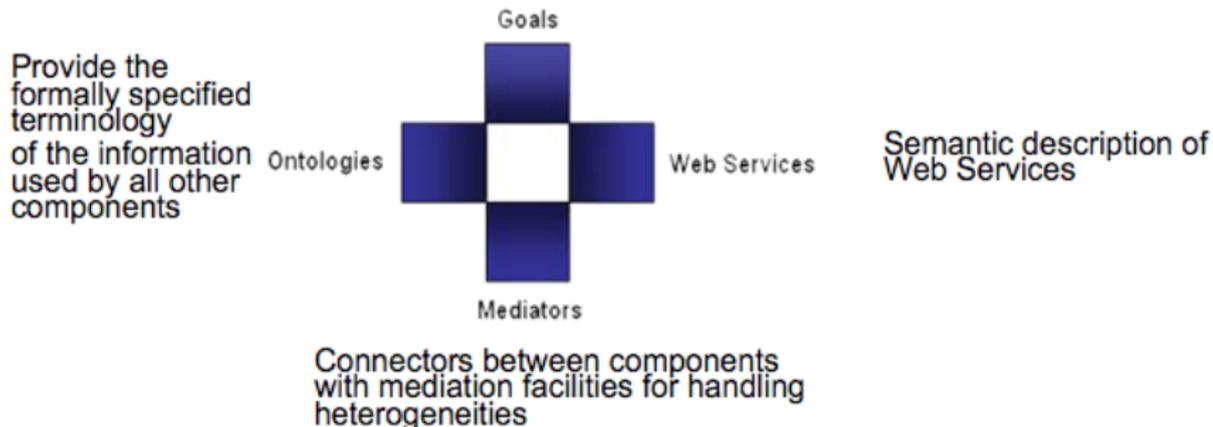
- ▶ WSMO is not an ontology in OWL, WSMO defines an own ontology language.
- ▶ Decouple provider and requester view.
- ▶ Decouple Interface from Implementation: distinguish between internal process and externally observable behavior.
- ▶ make mediation a first-class object



Still, many similarities with the OWL-S model.

WSMO top level concepts

Objectives that a client may have when consulting a Web Service



WSMO ontologies

- ▶ Define terminology (classes, attributes, axioms on terminology) used by a web service.
- ▶ Language: WSML
 - ▶ Ontology language in WSML closer to LP than OWL.
 - ▶ A more expressive language for expressing conditions, axioms, than OWL.
 - ▶ WSML (under development) is not only an ontology language but shall comprise a language for expressing all of WSMO.

Properties:

- ▶ Imported Ontologies: import existing ontologies where no heterogeneities arise
- ▶ Used mediators: OO Mediators (ontology import with terminology mismatch handling)
- ▶ “Standard” Ontology Notions: Concepts, Attributes, Relations, Functions, Instances, Axioms

WSMO services/goals

Define the provided/requested:

- ▶ capability
- ▶ interfaces

WSMO services/goals

Define the provided/requested:

- ▶ capability
- ▶ interfaces

Capability: comarable to OWL-S profile

Imported Ontologies

Used mediators OOMediators, WWMediators, WGMediators.

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 describe 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)

WSMO services/goals

Define the provided/requested:

- ▶ capability
- ▶ interfaces

Capability: comarable to OWL-S profile

Imported Ontologies

Used mediators OOMediators, WWMediators, WGMediators.

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.

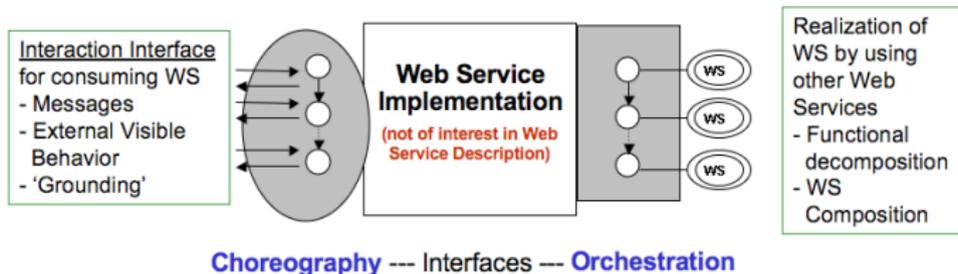
Post-conditions describe the result of the Web Service in relation to the input, and conditions on it.

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)

Interfaces: WSMO distinguishes *choreography* and *orchestration* interfaces

WSMO service/goal interfaces:

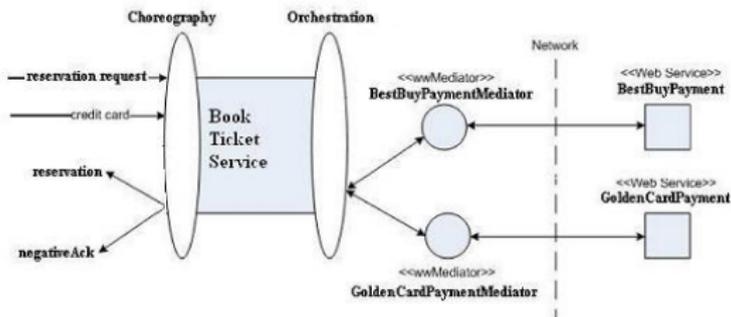
No workflow language but an automaton (abstract state machine) shall define the control and data flow. Final syntax still under discussion.



“Grounding” idea similar to OWL-S: input/output messages references to WSDL message-operation pair

WSMO service/goal interfaces:

No workflow language but an automaton (abstract state machine) shall define the control and data flow. Final syntax still under discussion.



A simple example.

- ▶ Choreography interface: externally observable behavior of the service
- ▶ Orchestration interface: which other services will be called by this service in order to fulfill its capability.

WSMO Services

Requester view, dual to Web service annotations:

- ▶ provide/guarantee non-functional properties
- ▶ import Ontologies
- ▶ use Mediators
- ▶ provide a Capability
- ▶ provide an Interface

WSMO Goals

Requester view, dual to Web service annotations:

- ▶ request non-functional properties
- ▶ import Ontologies
- ▶ use Mediators
- ▶ request a Capability
- ▶ request an Interface

WSMO Mediators (1/2):

Resolve mismatches in service interaction/between service annotations. Different levels of Heterogeneity:

- (1) Data Level: mediate heterogeneous Data Sources
- (2) Protocol/Process Level: mediate heterogeneous Communication Patterns and Business Processes.

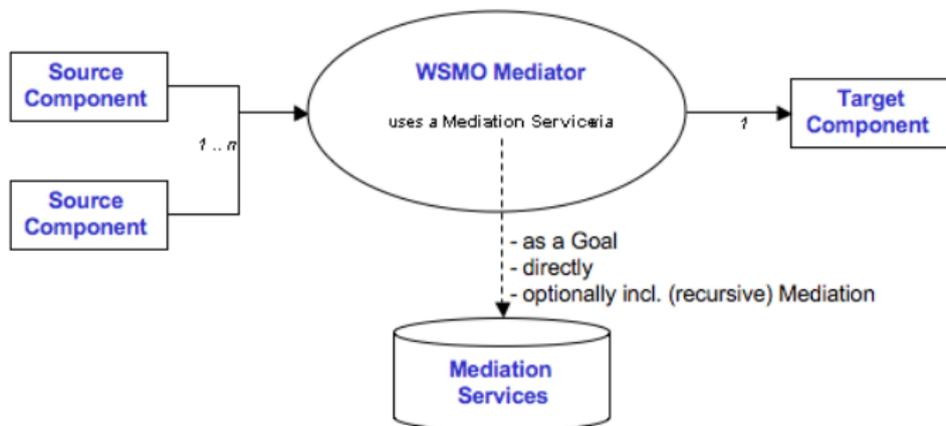
OOMediator: Define how concepts/relations can be mapped to another ontology. Mapping languages (under development) are basically powerful rule languages.

WGMediator: How can a service resolve a goal which does not “exactly” match? E.g. different interaction protocols require to split/merge messages, change order of messages, etc.

GGMediator: A goal can be a refinement of a more general goal, “Book a Trip” is more general than “Book a Flight”, etc

WSMO Mediators (2/2)

Properties:



SWSF

The Semantic Web Service Framework

<http://www.w3.org/Submission/SWSF/>

- ▶ Roots in OWL-S and PSL
- ▶ A first-order ontology for Semantic Web services, using the first-order notation of processes from PSL (ISO standard).
- ▶ remedies some weaknesses of OWL-S, by not being restricted to description logics.
- ▶ “grounding” problem not clearly addressed. No practical implementation efforts.
- ▶ also defines its own ontology and rule languages.

OWL-S, WSMO, SWSF

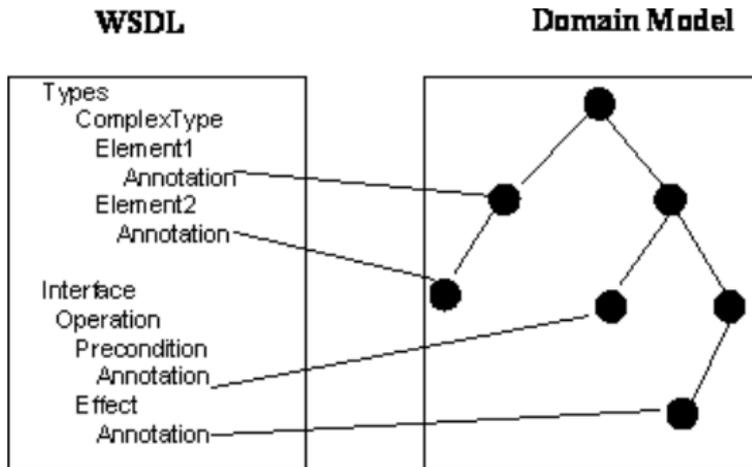
- ▶ “Heavy-weight” approaches
- ▶ Own languages, separate annotations
- ▶ still to a large extent research/academic (except big research projects with industry participation)
- ▶ not much emphasis so far to align with other WS-* standards (BPEL, WS-CDL, WS-Policy, WS-Security), except WSDL grounding.

A minimalistic approach: WSDL-S

- ▶ evolutionary and compatible upgrade of existing WS standards
- ▶ avoid duplication of what is already defined in WSDL
- ▶ minimal language commitment (OWL, UML, ? ...)
- ▶ Basically: embed what is needed from OWL-S profile directly in WSDL
- ▶ Why? Community is familiar with WSDL, provide a cautious extension.
- ▶ Claim: more practical approach for adoption

WSDL-S

<http://www.w3.org/Submission/WSDL-S/>



- ▶ define *service category*
- ▶ link operations to externally defined **operation ontology**
- ▶ link message types to externally defined concepts (e.g. defined in OWL)
- ▶ link operations to externally defined **preconditions** and **effects**

No commitment to formal language to be used, i.e. notions of match unclear.
For non-functional aspects, exploit existing WS-* standards. (not defined yet how),
e.g. "We are investigating how to represent QoS assertions using ontologies and rules
by extending the WS-Policy framework"

```

<?xml version="1.0" encoding="iso-8859-1"?>
<definitions name="PurchaseOrder"
targetNamespace="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/examples/purchaseOrder.wsdl"
xmlns="http://www.w3.org/2004/08/wsdl"
xmlns:tns="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/examples/purchaseOrder.wsdl"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsd1="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/examples/purchaseOrder.wsdl"
xmlns:wssem="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/examples/purchaseOrder.wsdl"
xmlns:POontology="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/ontologies/PurchaseOrder.owl"
xmlns:Rosetta="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/ontologies/rosetta.owl">
  <types>
    <xs:import namespace="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/examples/purchaseOrder.wsdl"
      schemaLocation="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/examples/WSSemantics.xsd"/>
    <xs:import namespace="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/examples/purchaseOrder.wsdl"
      schemaLocation="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/examples/POBilling.xsd" />
    <xs:import namespace="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/examples/purchaseOrder.wsdl"
      schemaLocation="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/examples/POItem.xsd" />
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/examples/purchaseOrder.wsdl"
      xmlns="http://lstdis.cs.uga.edu/projects/meteor-s/wsd1-s/examples/purchaseOrder.wsdl">
      <!--Semantic annotations for these complex types are given in their respective type
      definitions -->
      <xs:complexType name="processPurchaseOrderRequest">
        <xs:all>
          <xs:element name="billingInfo" type="xsd1:POBilling"/>
          <xs:element name="orderItem" type="xsd1:POItem"/>
        </xs:all>
      </xs:complexType>
      <!--Semantic annotation is added directly to leaf element -->
      <xs:element name="processPurchaseOrderResponse" type="xs:string"
        wssem:modelReference="POontology#OrderConfirmation"/>
    </xs:schema>
  </types>
  <interface name="PurchaseOrder">
    <!--Category is added as an extensible element of an interface-->
    <wssem:category name="Electronics" taxonomyURI="http://www.naics.com/" taxonomyCode="443112" />
    <operation name="processPurchaseOrder" pattern="wsdl:in-out"
      wssem:modelReference="Rosetta:RequestPurchaseOrder" >
      <input messageLabel="processPurchaseOrderRequest"
        element="tns:processPurchaseOrderRequest"/>
      <output messageLabel="processPurchaseOrderResponse"
        element="processPurchaseOrderResponse"/>
    <!--Precondition and effect are added as extensible elements on an operation-->
    <wssem:precondition name="ExistingAcctPrecond"
      wssem:modelReference="POontology#AccountExists"/>
    <wssem:effect name="ItemReservedEffect"
      wssem:modelReference="POontology#ItemReserved"/>
    </operation>
  </interface>
</definitions>

```

Comparison: Coverage of basic representational aspects

1. *General service classifications*: common to all approaches
2. *pre- and postconditions*: common to all approaches
3. *behavior/protocol* description of the service OWL-S, WSMO, SWSF allow to encode complex behavior, WSDL-S implicit, or e.g. by embedding into BPEL4WS
4. non-functional aspects (QoS, cost, availability, etc.) OWL-S, WSMO, SWSF provide extensible sets of non-functional properties, WSDL-S sees this out of scope

Comparison: Coverage of basic representational aspects

1. *General service classifications*: common to all approaches
 2. *pre- and postconditions*: common to all approaches
 3. *behavior/protocol* description of the service OWL-S, WSMO, SWSF allow to encode complex behavior, WSDL-S implicit, or e.g. by embedding into BPEL4WS
 4. non-functional aspects (QoS, cost, availability, etc.) OWL-S, WSMO, SWSF provide extensible sets of non-functional properties, WSDL-S sees this out of scope
- ▶ Mediators: Own concept in WSMO, in OWL-S and SWSF not treated separately, but just as special kind of service.

Comparison: Coverage of basic representational aspects

1. *General service classifications*: common to all approaches
 2. *pre- and postconditions*: common to all approaches
 3. *behavior/protocol* description of the service OWL-S, WSMO, SWSF allow to encode complex behavior, WSDL-S implicit, or e.g. by embedding into BPEL4WS
 4. non-functional aspects (QoS, cost, availability, etc.) OWL-S, WSMO, SWSF provide extensible sets of non-functional properties, WSDL-S sees this out of scope
- ▶ Mediators: Own concept in WSMO, in OWL-S and SWSF not treated separately, but just as special kind of service.
 - ▶ Goal/requester view: Motivation to in WSMO separate concerns, goals/requests not treated in WSDL-S. Main issues:
 - ▶ How is a request/query to be formulated?
 - ▶ What are the related notions of “match”?
- a certain degree of language committment seems necessary

Standardization Activities

- ▶ W3C Semantic Annotations for WSDL Working Group
 - ▶ Charter currently being drafted
 - ▶ WSDL-S a likely starting point
- ▶ W3C SWS IG <http://www.w3.org/2005/09/sws-ig-charter>
- ▶ OASIS Semantic Web Services Architecture and Information Model

Issues/Connections

- ▶ No agreement yet in the community on formal underpinnings.
- ▶ Connections to multiple fields in AI:
 - ▶ Formal languages, reasoning (Description Logics Reasoning, Query Answering, Theorem Proving, Logic Programming)
 - ▶ Reasoning about processes, dynamics (bi-simulation, planning)
 - ▶ Multi-agent systems (probably similar conceptual frameworks, problems)
- ▶ Strong industry interest!

Summary

This talk was restricted to representational issues, not the methods to be applied:

Summary

This talk was restricted to representational issues, not the methods to be applied:

- ▶ Outline application area

Summary

This talk was restricted to representational issues, not the methods to be applied:

- ▶ Outline application area
- ▶ “Service annotation” as the next step after annotation of static data on the web (Semantic web)
- ▶ Proposals/Approaches

Summary

This talk was restricted to representational issues, not the methods to be applied:

- ▶ Outline application area
- ▶ “Service annotation” as the next step after annotation of static data on the web (Semantic web)
- ▶ Proposals/Approaches
- ▶ Representational aspects

Summary

This talk was restricted to representational issues, not the methods to be applied:

- ▶ Outline application area
- ▶ “Service annotation” as the next step after annotation of static data on the web (Semantic web)
- ▶ Proposals/Approaches
- ▶ Representational aspects

You want to know more?

Outlook

- ▶ You want about how methodology/methods from your course can be deployed here?
- ▶ Discuss concrete use cases?
- ▶ Investigate WS-* standards in detail?
- ▶ Get your hands dirty in programming WS? ;-)
- ▶ Help in developing intelligent Web, intelligent Web Services?

Outlook

- ▶ You want about how methodology/methods from your course can be deployed here?
- ▶ Discuss concrete use cases?
- ▶ Investigate WS-* standards in detail?
- ▶ Get your hands dirty in programming WS? ;-)
- ▶ Help in developing intelligent Web, intelligent Web Services?

Posibilidades:

- ▶ Proyectos fin de carrera!
- ▶ Becas para proyectos concretos posible
- ▶ Colaboraciones internacionales! (DERI, W3C, TU Viena, Univ.Calabria, etc.)

Outlook

- ▶ You want about how methodology/methods from your course can be deployed here?
- ▶ Discuss concrete use cases?
- ▶ Investigate WS-* standards in detail?
- ▶ Get your hands dirty in programming WS? ;-)
- ▶ Help in developing intelligent Web, intelligent Web Services?

Posibilidades:

- ▶ Proyectos fin de carrera!
- ▶ Becas para proyectos concretos posible
- ▶ Colaboraciones internacionales! (DERI, W3C, TU Viena, Univ.Calabria, etc.)

Expectaciones:

- ▶ Motivación, para trabajar y aprender en un area desarrollando rapido (muchas especificaciones largos solo online, ...)
- ▶ Desafío: Cobinación de aspectos muy practicos con teoria y IA!
- ▶ SOAs son el futuro, hay mucho potencial!

Otros asignaturas

- ▶ Otoño: Axel Polleres, David Pearce “Métodos Avanzados de Razonamiento para Tecnologías del Conocimiento y Web Semántica”
- ▶ Primavera: Axel Polleres “Next Web Generation” (libre elección, en Inglés)

Thank you for your attention!