ID2208 Programming Web Services

Semantic Web Services

Mihhail Matskin:
http://people.kth.se/~misha/ID2208/

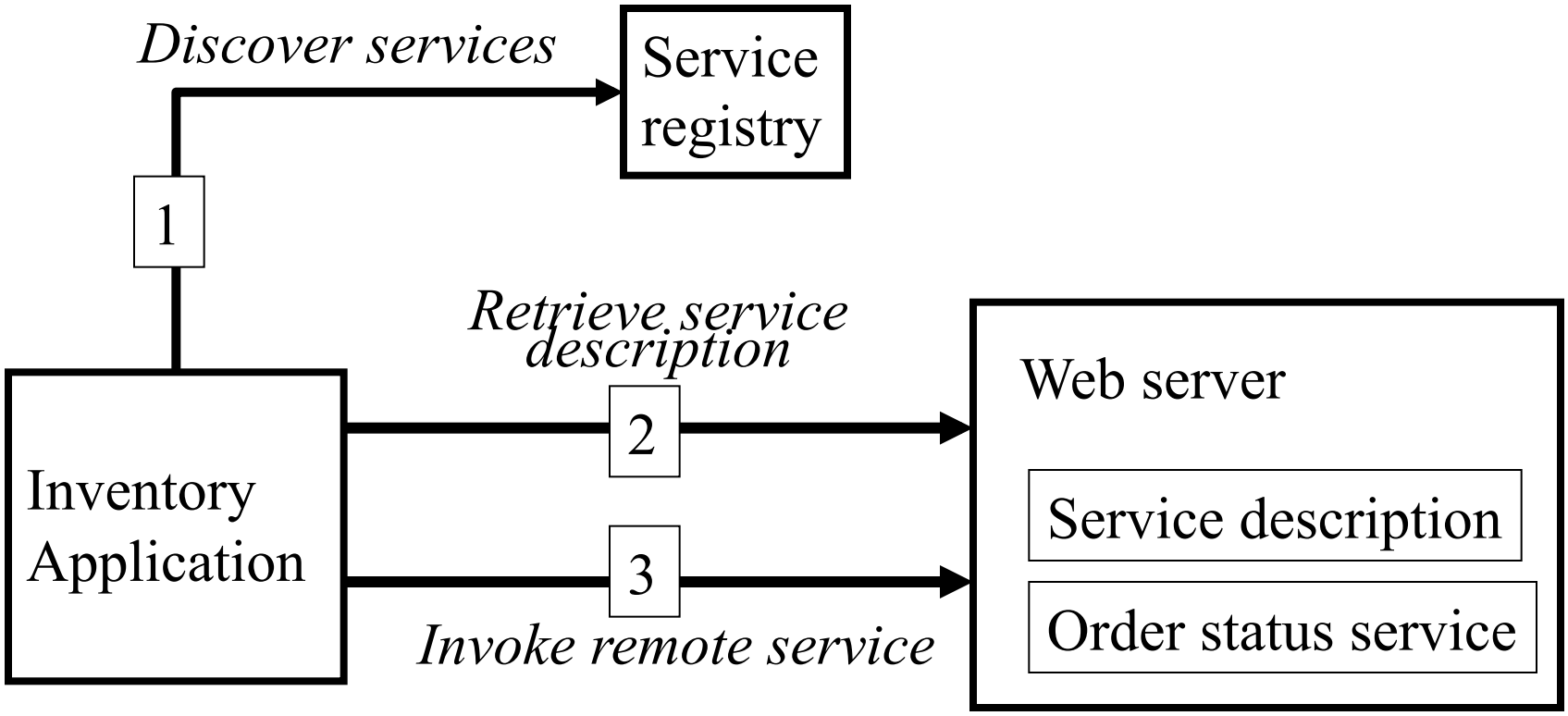
Spring 2016

Content

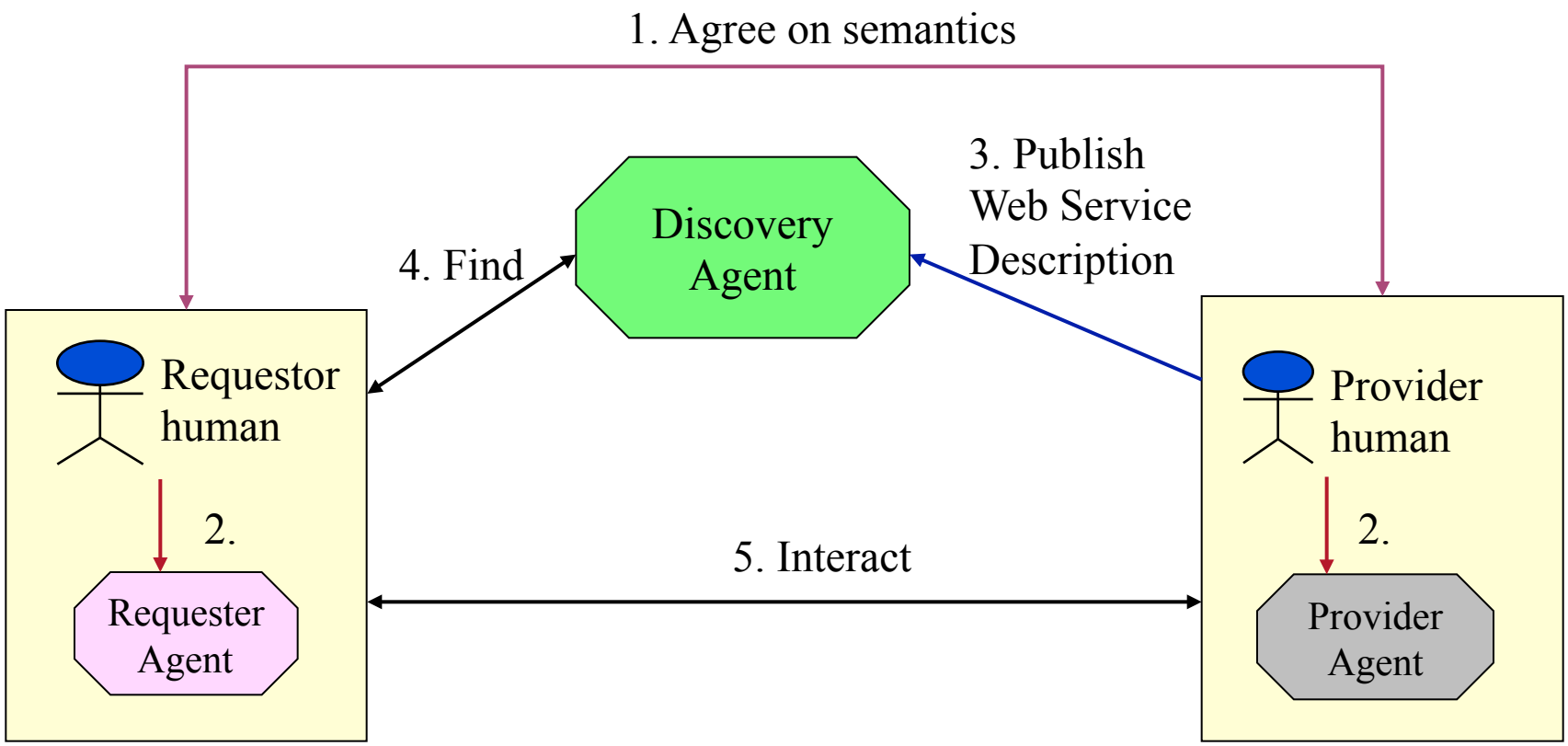
Semantic Web Services

- Semantic Web
- RDF Model
- Service ontology – OWL-S
- Profile and process
- OWL-S and WSDL

Web Services vision (automated web)



W3C: Web Services Architecture



Adapted from [W3C, 2003]

Limitations of current Web services

- No computer understandable semantics
- Discovery is based on keywords
- Interoperation is limited
- Limited run-time discovery, composition, negotiation, monitoring
- Limited mechanisms for describing business relations, agreements, contracts

Why do we need Semantics in Web service?

- Service location: syntactically different terms can be used by requestor and provider for the same service/notions
- Service invocation: constructing correct messages based on published service interfaces
- Results understanding: interpreting results of service invocation
- Service composition: constructing complex composite services by combining results of different services

WWW

- The current World Wide Web (WWW) is the syntactic Web
  - structure of the content has been presented
  - while the content itself is non-understandable to computers.
- The WWW still cannot fulfill the interoperation among various applications without some pre-existing, human-created agreements somewhere in-house or outside of the Web.

Semantic Web

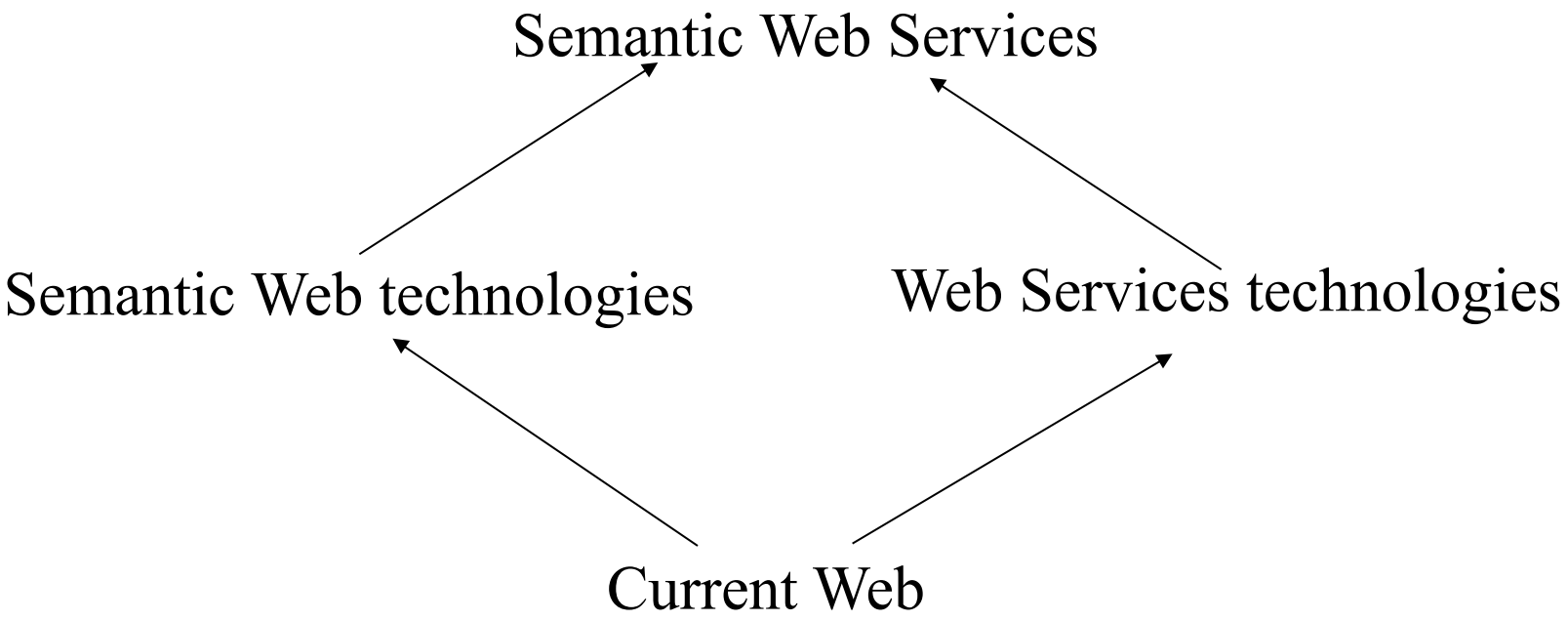
"The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. It is based on the idea of having data on the Web defined and linked such that it can be used for more effective discovery, automation, integration, and reuse across various applications "

Berners-Lee, Hendler, Lassila

Web of data

- Transferring content between web *applications* is still difficult even thought the use of the Extensible Markup Language (XML) helps a lot
- The problem of effectively exchanging data doesn't go away. For every pair of applications someone has to create an "XML to XML bridge."
- A capability beyond that offered by XML-Schema is needed if we are to provide mapping capabilities between divergent schemas.
- A fundamental component of the Semantic Web is the Resource Description Framework (RDF)

Future Web Services



Semantic Web Services

- Provide a Web services infrastructure with semantic presentation component/semantic information
- Provide ontology to describe Web services

RDF Basics

- Resource Definition Framework (RDF) is a language for representing information (including metadata) about resources on the WWW
- It is particularly intended for representing metadata about Web resources
- RDF is intended for situations where information should be processed by applications
- RDF uses Web identifiers (*URIs*), and describes resources in terms of simple properties and property values.

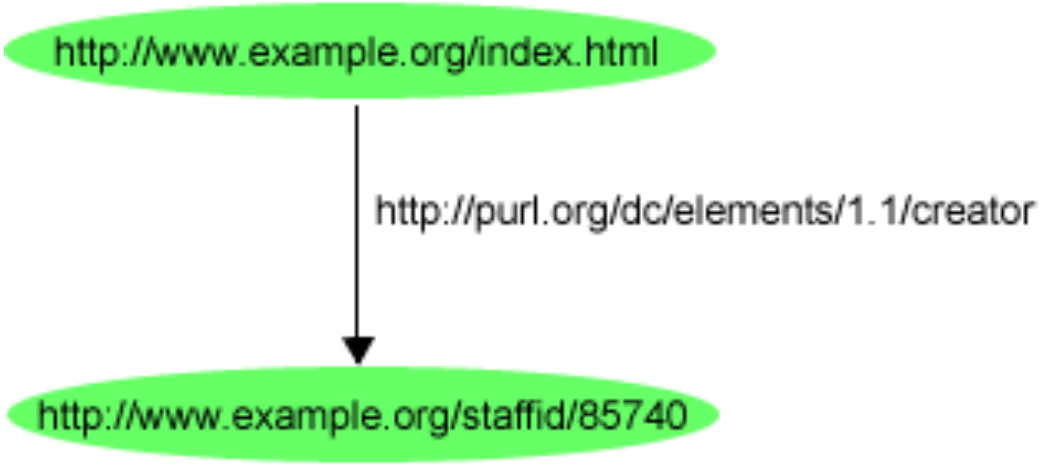
RDF Model

- URIs is basic way of referencing in RDF:

http://www.example.org/index.html has a **creator** whose value is **John Smith**

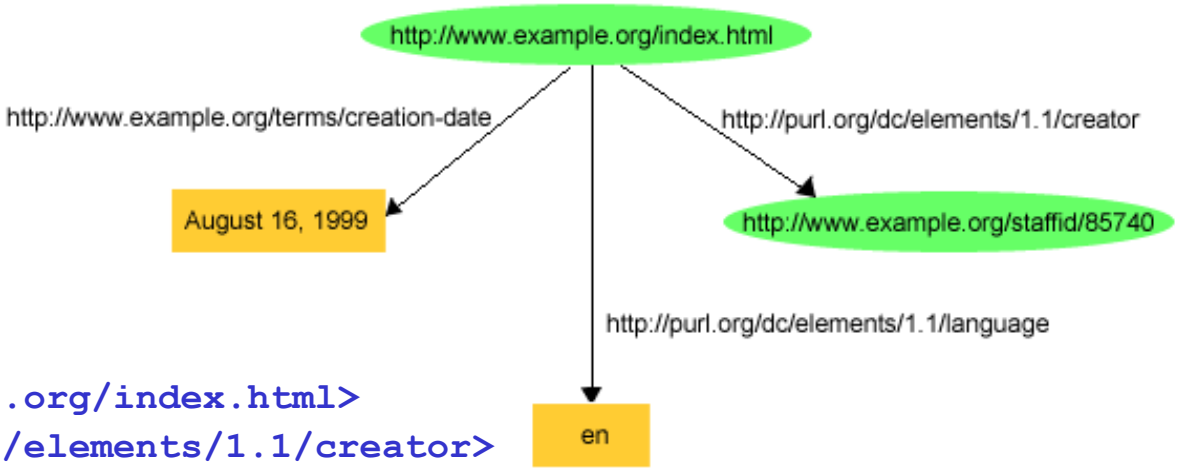
  - a subject http://www.example.org/index.html
  - a predicate http://purl.org/dc/elements/1.1/creator
  - an object http://www.example.org/staffid/85740

RDF Model



RDF model

http://www.example.org/index.html has a creator whose value is John Smith
http://www.example.org/index.html has a creation-date whose value is August 16, 1999
http://www.example.org/index.html has a language whose value is English

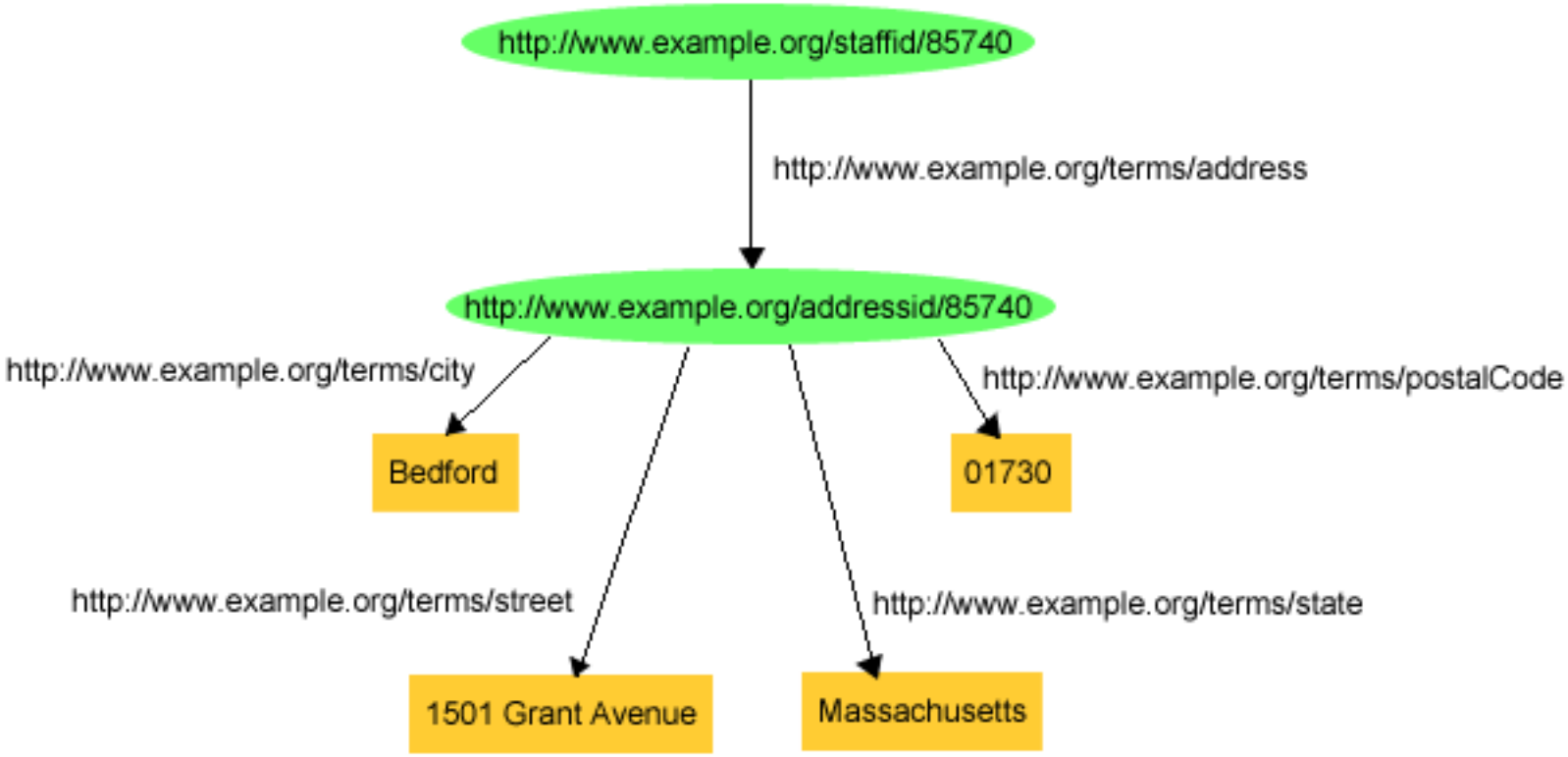



```
<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/creator>
<http://www.example.org/staffid/85740> .

<http://www.example.org/index.html>
<http://www.example.org/terms/creation-date>
"August 16, 1999" .

<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/language>
"en" .
```

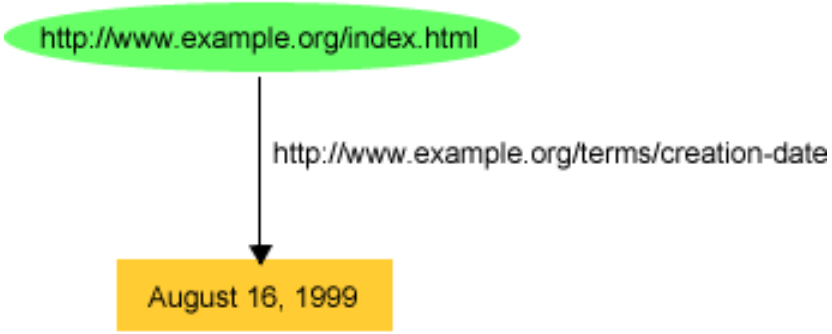
RDF model (structures)



RDF Typing

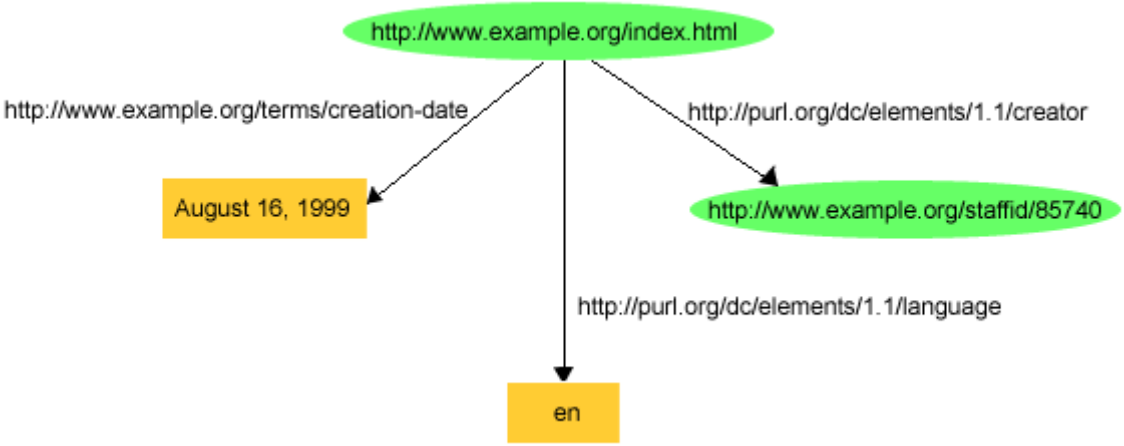
RDF/XML

http://www.example.org/index.html has a **creation-date** whose value is **August 16, 1999**




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:exterms="http://www.example.org/terms/">
   <rdf:Description rdf:about="http://www.example.org/index.html">
        <exterms:creation-date>
                August 16, 1999
        </exterms:creation-date>
   </rdf:Description>
 </rdf:RDF>
```

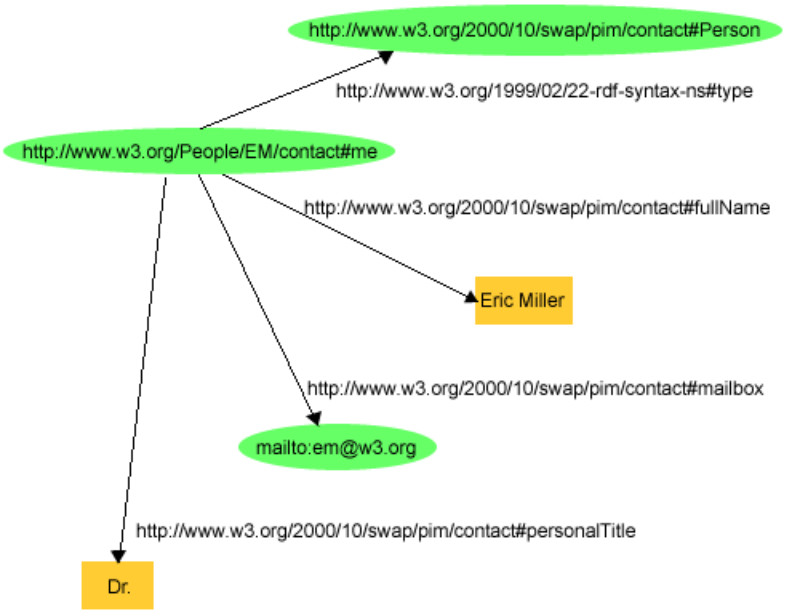
RDF/XML




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:exterms="http://www.example.org/terms/">
   <rdf:Description rdf:about="http://www.example.org/index.html">
        <exterms:creation-date>August 16, 1999</exterms:creation-date>
        <dc:language>en</dc:language>
        <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
   </rdf:Description>
</rdf:RDF>
```

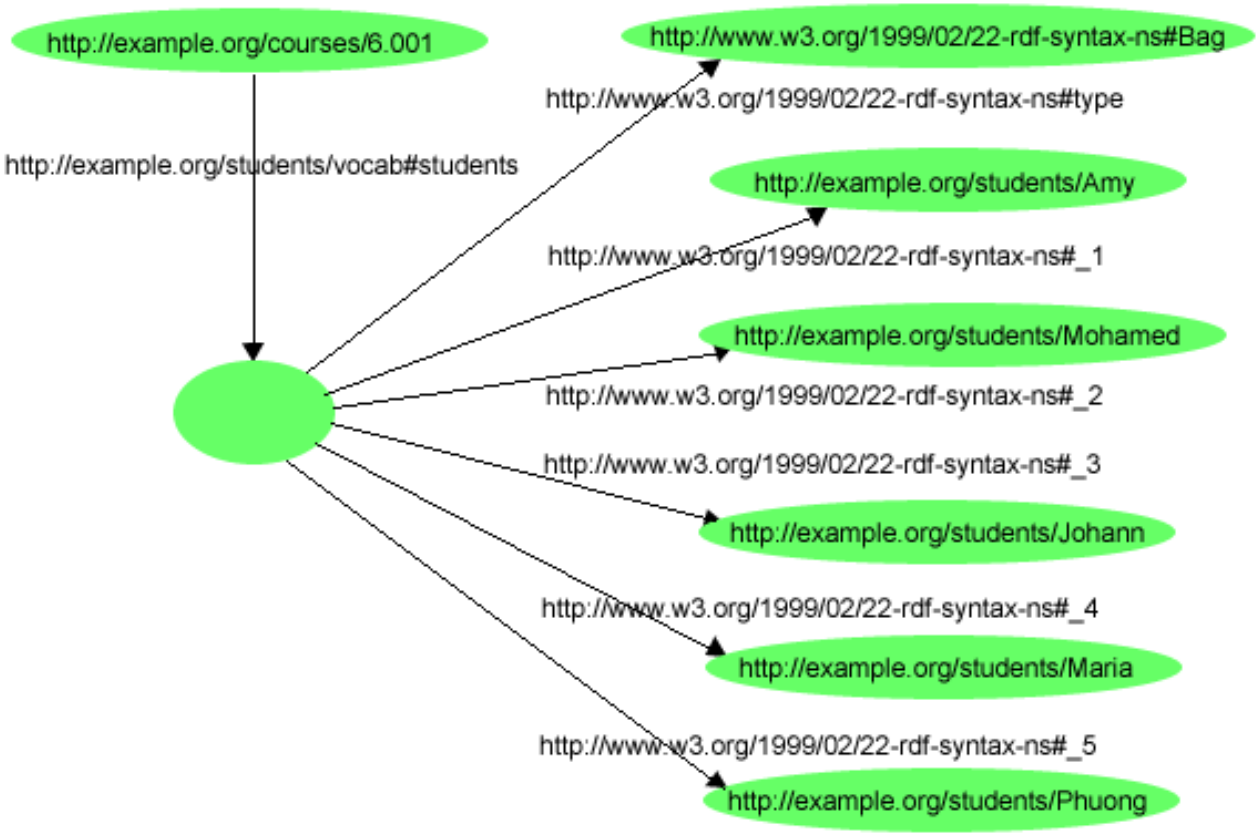
RDF/XML

"there is a Person identified by http://www.w3.org/People/EM/contact#me,
whose name is Eric Miller, whose email address is
em@w3.org, and whose title is Dr."




```
<?xml version="1.0"
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
   <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
        <contact:fullName>Eric Miller</contact:fullName>
        <contact:mailbox rdf:resource="mailto:em@w3.org"/>
        <contact:personalTitle>Dr.</contact:personalTitle>
   </contact:Person>
</rdf:RDF>
```

RDF containers

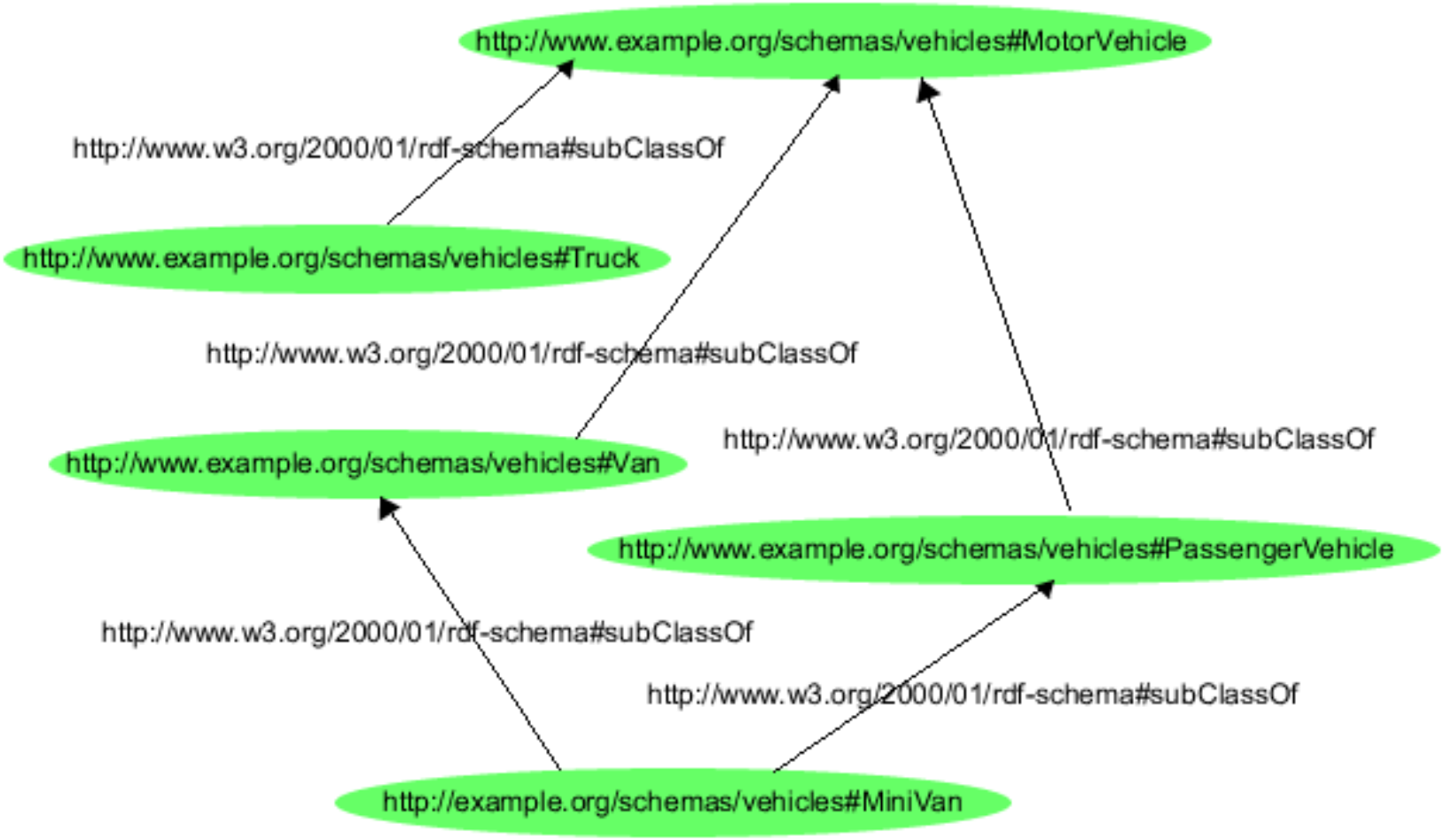


"Course 6.001 has the students Amy, Mohamed, Johann, Maria, and Phuong",

RDF containers /XML

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">
   <rdf:Description rdf:about="http://example.org/courses/6.001">
    <s:students>
        <rdf:Bag>
          <rdf:li rdf:resource="http://example.org/students/Amy"/>
          <rdf:li rdf:resource="http://example.org/students/Mohamed"/>
          <rdf:li rdf:resource="http://example.org/students/Johann"/>
          <rdf:li rdf:resource="http://example.org/students/Maria"/>
          <rdf:li rdf:resource="http://example.org/students/Phuong"/>
        </rdf:Bag>
    </s:students>
   </rdf:Description>
</rdf:RDF>
```

RDF Schema



RDF Schema / XML

```
<?xml version="1.0"?>
…
 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
          xml:base="http://example.org/schemas/vehicles">
  <rdfs:Class rdf:ID="MotorVehicle"/>
  <rdfs:Class rdf:ID="PassengerVehicle">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Truck">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Van">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="MiniVan">
    <rdfs:subClassOf rdf:resource="#Van"/>
    <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
  </rdfs:Class>
</rdf:RDF>
```

Instances

```
<?xml version="1.0"?>
   <rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:ex="http://example.org/schemas/vehicles#"
      xml:base="http://example.org/things">
     <ex:MotorVehicle rdf:ID="companyCar"/>
</rdf:RDF>
```

Describing properties

```
<rdf:Property rdf:ID="registeredTo">
   <rdfs:domain rdf:resource="#MotorVehicle"/>
   <rdfs:range rdf:resource="#Person"/>
</rdf:Property>
<rdf:Property rdf:ID="rearSeatLegRoom">
   <rdfs:domain rdf:resource="#PassengerVehicle"/>
   <rdfs:range rdf:resource="&xsd;integer"/>
</rdf:Property>
<rdfs:Class rdf:ID="Person"/>
<rdfs:Datatype rdf:about="&xsd;integer"/ >
```

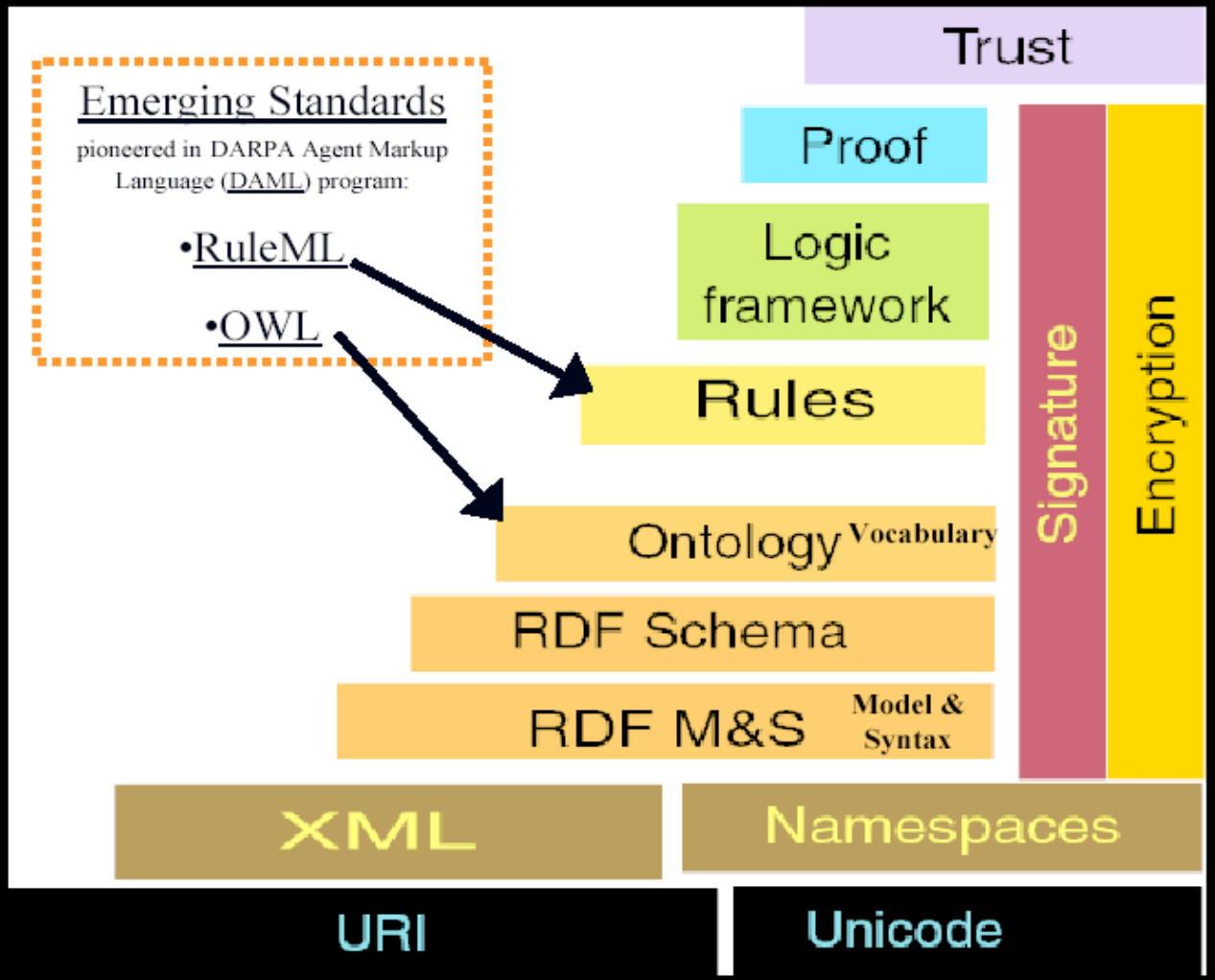
Vehicle Schema

```
<?xml version="1.0"?>
 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xml:base="http://
    example.org/schemas/vehicles">
 <rdfs:Class rdf:ID="MotorVehicle"/>
 <rdfs:Class rdf:ID="PassengerVehicle">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
 </rdfs:Class>
 <rdfs:Class rdf:ID="Truck">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
 </rdfs:Class>
 <rdfs:Class rdf:ID="Van">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
 </rdfs:Class>
 <rdfs:Class rdf:ID="MiniVan">
    <rdfs:subClassOf rdf:resource="#Van"/>
    <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
 </rdfs:Class>
 <rdfs:Class rdf:ID="Person"/>
 <rdfs:Datatype rdf:about="&xsd;integer"/>
 <rdf:Property rdf:ID="registeredTo">
    <rdfs:domain rdf:resource="#MotorVehicle"/>
    <rdfs:range rdf:resource="#Person"/>
 </rdf:Property>
 <rdf:Property rdf:ID="rearSeatLegRoom">
    <rdfs:domain rdf:resource="#PassengerVehicle"/>
    <rdfs:range rdf:resource="&xsd;integer"/>
 </rdf:Property>
 <rdf:Property rdf:ID="driver">
    <rdfs:domain rdf:resource="#MotorVehicle"/>
    <rdfs:range rdf:resource="#Person"/>
 </rdf:Property>
 <rdf:Property rdf:ID="primaryDriver">
    <rdfs:subPropertyOf rdf:resource="#driver"/>
 </rdf:Property>
</rdf:RDF>
```

Instance

```
<?xml version="1.0"?>
…
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://example.org/schemas/vehicles#"
         xml:base="http://example.org/things">
   <ex:PassengerVehicle rdf:ID="johnSmithsCar">
      <ex:registeredTo rdf:resource=
                "http://www.example.org/staffid/85740"/>
      <ex:rearSeatLegRoom rdf:datatype="&xsd;integer">
                127
      </ex:rearSeatLegRoom>
      <ex:primaryDriver rdf:resource=
                "http://www.example.org/staffid/85740"/>
   </ex:PassengerVehicle>
</rdf:RDF>
```

Semantic Web Stack



[Diagram http://www.w3.org/DesignIssues/diagrams/sw-stack-2002.png is courtesy Tim Berners-Lee]

5/23/2003

OWL Ontology language

- OWL adds more vocabulary for describing properties and classes among others: relations between classes (e.g. disjointness), cardinality (e.g. "exactly one‟ instance), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.
- Different types of constraints can be expressed: **`equivalentProperty , inverseOf, TransitiveProperty, SymmetricProperty`** etc.
- A logical reasoning can be applied to the ontology. The logic which is used is description logic (or some other logics - F-Logic).

OWL

```
<owl:Object property rdf:=“livesIn”>
   <rdfs:domain rdf:resource=‘#animal’/>
   <rdfs:range rdf:resource=‘#locale’/>
   <rdfs:subPropertyOf rdf:resource=‘#hasHabitat’/>
   <owl:inverseOf rdf:resource=‘#hasDenizen’/>
   <owl:equivalentProperty rdf:resource=‘#hasHome’/>
</owl:objectProperty>
```

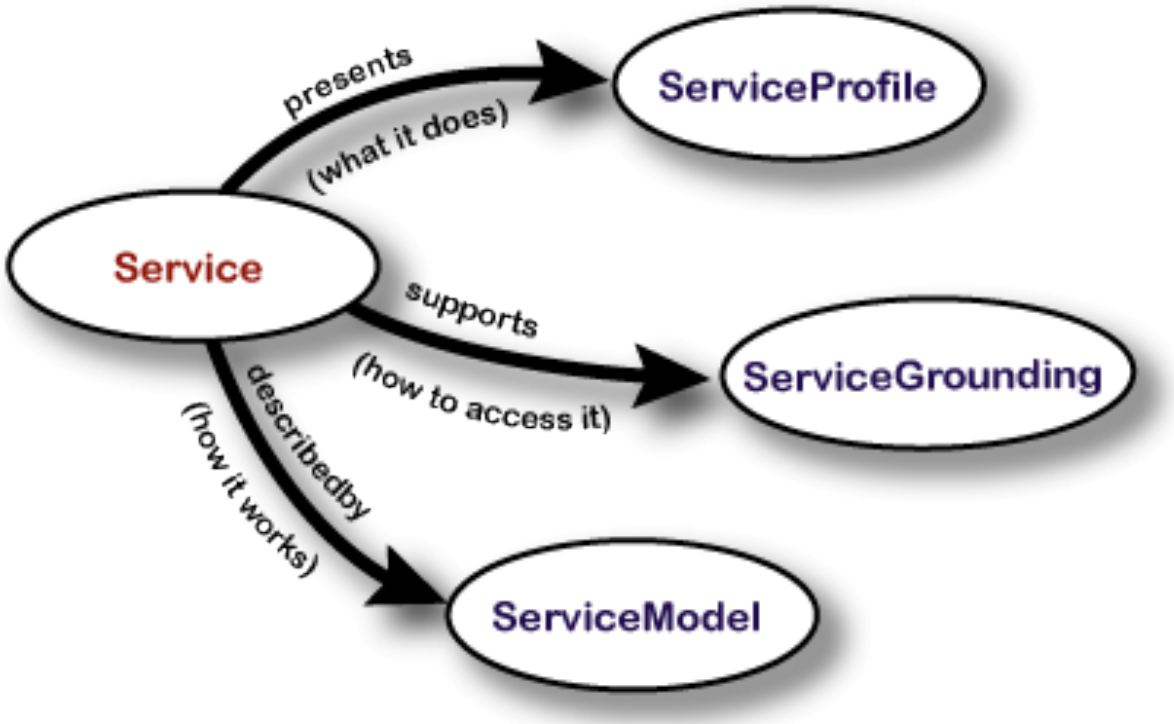
OWL Ontology Language

- *OWL Lite* supports those users primarily needing a classification hierarchy and simple constraints.
- *OWL DL* supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time).
- *OWL Full* is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees.

OWL-S

OWL-S ontology is a *language* for describing services. It provides a standard vocabulary that can be used together with the other aspects of the OWL description language to create service descriptions

Top level of the service ontology



From: **The OWL Services Coalition**

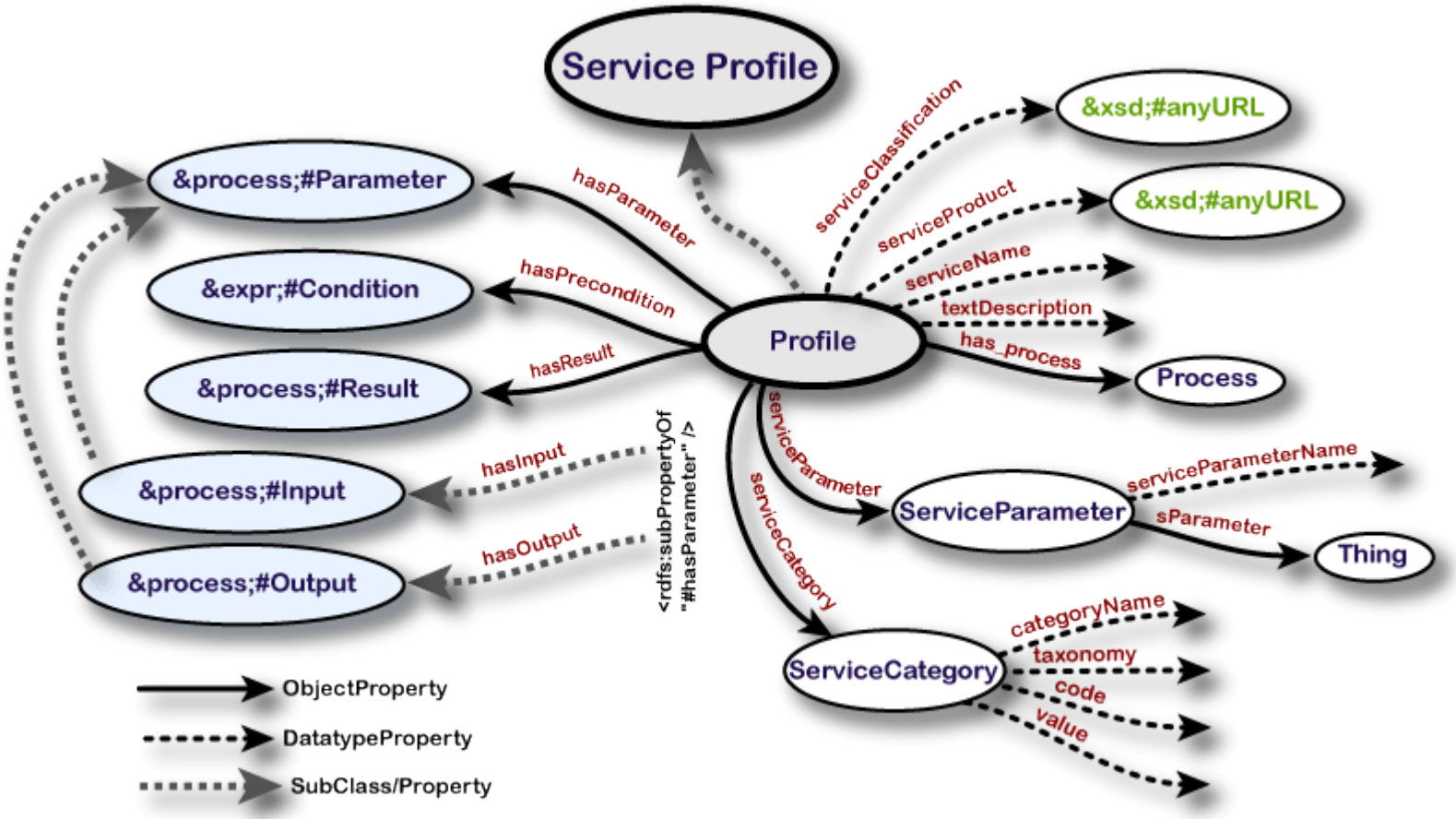
Service Profile and Process Model

- They are two different representations of the same service
- The input, output, precondition, and effects (IOPEs) of one are reflected in the IOPEs of the other

Functionality description

- OWL-S Profile represents two aspects of the functionality of the service:
  - Information transformation (input/output)
  - State change (preconditions/effects)
- IOPE Properties in Profile:
  - **`hasInput`** - ranges over an Input instance.
  - **`hasOutput`** - ranges over instances of type **`Output`**, as defined in the Process ontology
  - **`hasPrecondition`** specifies one of the preconditions of the service and ranges over a **`Precondition`** instance defined according to the schema in the **`Process`** ontology.
  - **`hasResult`** specifies one of the effects of the service.

Selected classes and properties of the Profile



From: **The OWL Services Coalition**

Services as Processes

- Service can be viewed as a process
- Process is not an executable program but rather a specification
- Process can be atomic or composite
- Process can
  - generate and return some new information based on information it is given and the world state (information transformation)
  - produce a change in the world (state change)

Process Parameters and Results

- **Participants:**
  - **`theClient`**, the agent from whose point of view the process is described.
  - **`theServer`**, the principal element of the service that the client deals with
- **Inputs and Outputs:**
  - specify the data transformation produced by the process
- **Preconditions:**
  - process cannot be performed successfully unless the precondition is true
- **Conditioning Outputs and Effects** a process model can describe the result in terms of properties:
  - **`inCondition`** - specifies the condition under which this result occurs
  - **`withOutput`** and **`hasEffect`** state what ensures when the condition is true
  - **`hasResultVar`** –variables that are bound in the **`inCondition`**

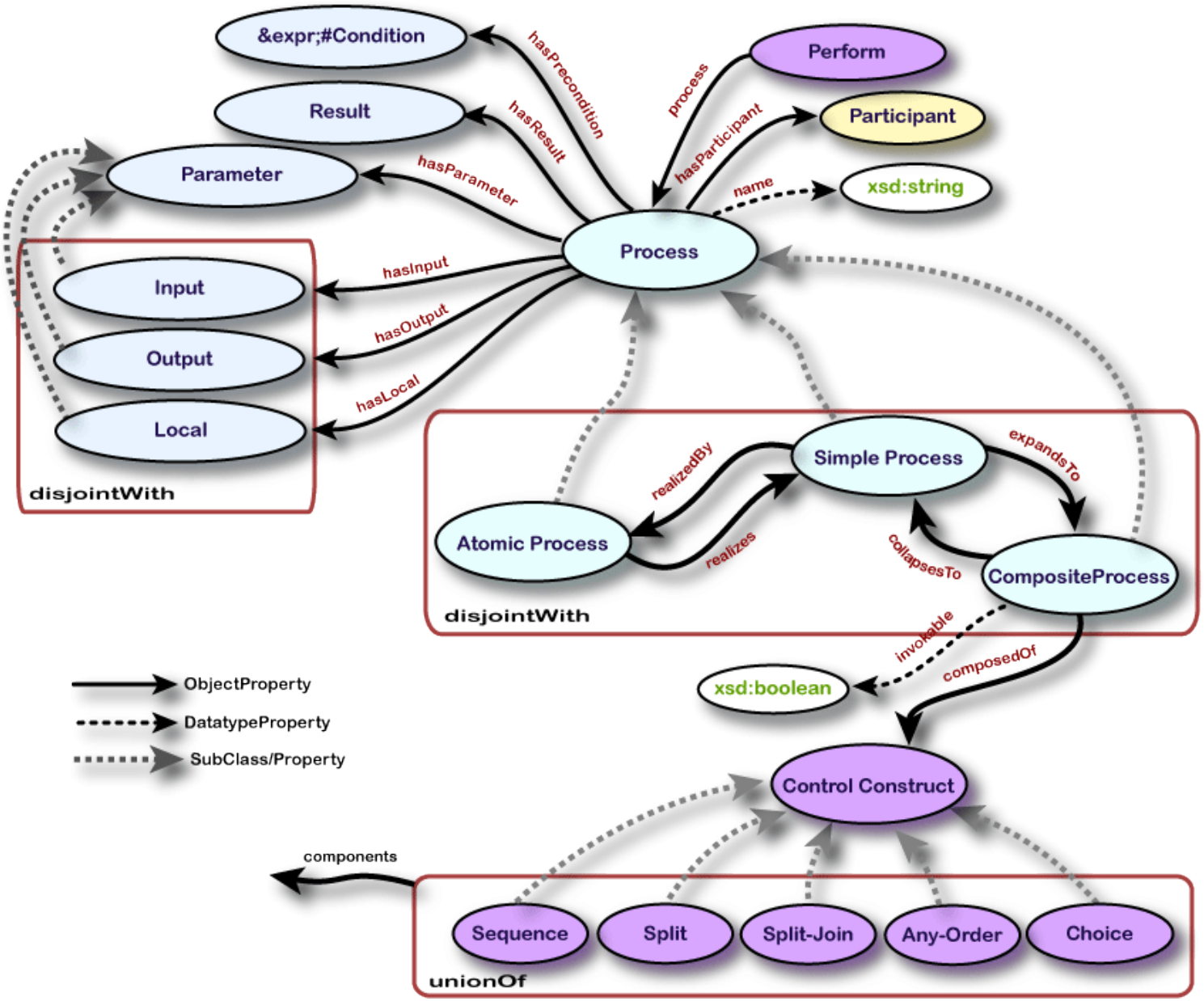
Process description

```
<pr:AtomicProcess rdf:ID="Purchase">
  <pr:hasInput>
        <pr:Input rdf:ID="ObjectPurchased"/>
  </pr:hasInput>
  <pr:hasInput>
        <pr:Input rdf:ID="PurchaseAmt"/>
  </pr:hasInput>
  <pr:hasInput>
        <pr:Input rdf:ID="CreditCard"/>
  </pr:hasInput>
  <pr:hasOutput>
        <pr:Output rdf:ID="ConfirmationNum"/>
  </pr:hasOutput>
  <pr:hasResult>
    <pr:Result>
      <pr:hasResultVar>
          <pr:ResultVar rdf:ID="CreditLimH">
              <pr:parameterType rdf:resource="&ecom;#Dollars"/>
          </pr:ResultVar>
      </pr:hasResultVar>
      <pr:inCondition expressionLanguage="&expr;#KIF"
                      rdf:dataType="&xsd;#string">
          (and (current-value (credit-limit ?CreditCard) ?CreditLimH)
               (>= ?CreditLimH ?purchaseAmt))
      </pr:inCondition>
      <pr:withOutput>
          <pr:OutputBinding>
             <pr:toParam rdf:resource="#ConfirmationNum"/>
             <pr:valueFunction rdf:parseType="Literal">
                  <cc:ConfirmationNum xsd:datatype="&xsd;#string"/>
             </pr:valueFunction>
          </pr:OutputBinding>
      </pr:withOutput>
```

Process description (cntd.)

```
        <pr:hasEffect expressionLanguage="&expr;#KIF" rdf:dataType="&xsd;#string">
            (and (confirmed (purchase ?purchaseAmt) ?ConfirmationNum)
                 (own ?objectPurchased)
                 (decrease (credit-limit ?CreditCard) ?purchaseAmt))
        </pr:hasEffect>
    </pr:Result>
    <pr:Result>
        <pr:hasResultVar>
            <pr:ResultVar rdf:ID="CreditLimL">
                  <pr:parameterType rdf:resource="&ecom;#Dollars"/>
            </pr:ResultVar>
        </pr:hasResultVar>
        <pr:inCondition expressionLanguage="&expr;#KIF" rdf:dataType="&xsd;#string">
            (and (current-value (credit-limit ?CreditCard) ?CreditLimL)
                 (< ?CreditLimL ?purchaseAmt))
        </pr:inCondition>
        <pr:withOutput rdf:resource="&ecom;failureNotice"/>
           <pr:OutputBinding>
             <pr:toParam rdf:resource="#ConfirmationNum"/>
             <pr:valueData rdf:parseType="Literal">
                <drs:Literal>
                   <drs:litdefn xsd:datatype="&xsd;#string">00000000</drs:litdefn>
                </drs:Literal>
             </process:valueData>
           </pr:OutputBinding>
        </pr:withOutput>
    </pr:Result>
  </pr:hasResult>
</pr:AtomicProcess>
```

Top level of the process ontology



From: **The OWL Services Coalition**

Data Flow and Parameter Bindings

- The input to one process component can be obtained as one of the outputs of a preceding step (this is a type of dataflow)
- Specifying which component's output becomes output of the composite process is also a data-flow specification
- The source of a datum is identified when the user of the datum is declared

•Input I1 of the overall process CP is used as input I11 of S1, after adding 1.
•Input I12 of S1 is a constant, the string "Academic".
•Output O11 of S1 is used as input I21 of S2.
•The maximum of 0 and output O21 of S2, times π, is used as output O1 of CP.

I11(S1)	*comes from*	incr(I1(CP))
I12(S1)	=	"Academic"
I21(S2)	*comes from*	O11(S1)
O1(CP)	*comes from*	$\pi \times \max(0, \text{O21(S2)})$

Process

```
<pr:CompositeProcess rdf:ID="CP">
 <pr:hasInput rdf:ID="I1"/>
 <pr:hasOutput rdf:ID="O1"/>
 <pr:composedOf>
 <pr:Sequence rdf:ID="CP">
  <pr:components rdf:parseType="Collection">
   <pr:Perform rdf:ID="Step1">
    <pr:process rdf:resource="&aux;#S1"/>
    <pr:hasDataFrom>
     <pr:InputBinding>
      <pr:theParam rdf:resource="&aux;#I11"/>
      <pr:valueFunction expressionLanguage="&drs;" rdf:parseType="Literal">
       <drs:Functional_term>
        <drs:term_function rdf:resource="&arith;#incr"/>
        <drs:term_args rdf:parseType="Collection">
         <pr:valueOf>
          <pr:theParam rdf:resource="#I1"/>
          <pr:fromProcess rdf:resource="#TheParentPerform"/>
         </pr:valueOf>
        </drs:term_args>
       </drs:Functional_term>
      </pr:valueFunction>
      </pr:InputBinding>
      <pr:InputBinding>
       <pr:theParam rdf:resource="&aux;#I12"/>
       <pr:valueData xsd:datatype="&xsd;#string">Academic</pr:valueData>
      </pr:InputBinding>
     </pr:hasDataFrom>
    </pr:Perform>
    <pr:Perform rdf:ID="Step2">
     <pr:process rdf:resource="&aux;#S2"/>
     <pr:hasDataFrom>
      <pr:Binding>
       <pr:theParam rdf:resource="&aux;#I21"/>
       <pr:valueSource>
        <pr:ValueOf>
         <pr:theParam rdf:resource="#O11"/>
         <pr:fromProcess rdf:resource="#Step1"/>
        </pr:ValueOf>
       </pr:valueSource>
      </pr:Binding>
     </pr:hasDataFrom>
    </pr:Perform>
   </pr:components>
```

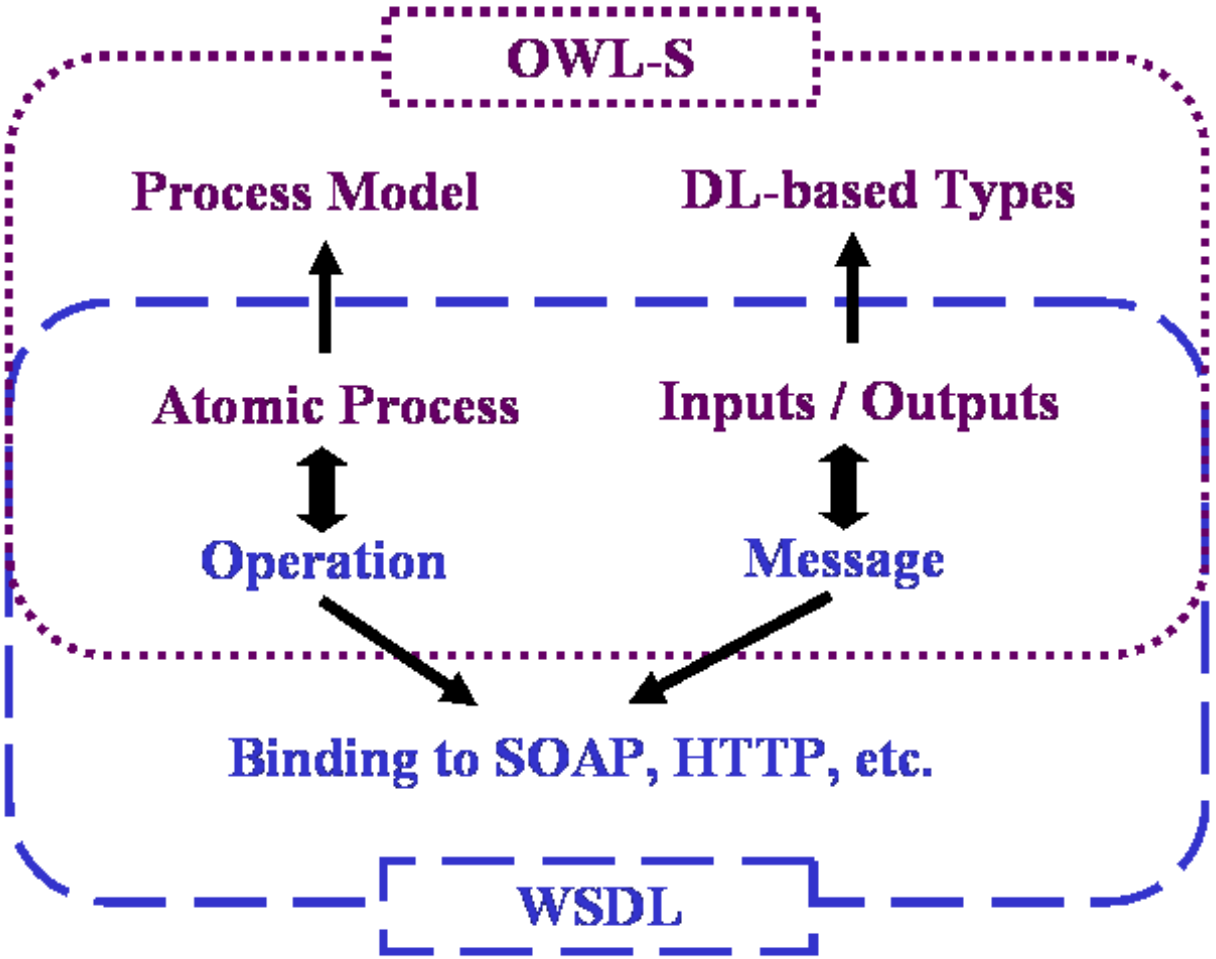
Process (cntd.)

```
<pr:Produce>
  <pr:producedBinding>
   <pr:OutputBinding>
    <pr:theParam rdf:resource="#O1"/>
    <pr:valueFunction expressionLanguage="&drs;" rdf:parseType="Literal">
     <drs:Functional_term>
      <drs:term_function rdf:resource="&arith;#times"/>
      <drs:term_args rdf:parseType="Collection">
       <xsd:Integer rdf:datatype="&xsd;#Float">3.14159</xsd:Float>
       <drs:Functional_term>
        <drs:term_function rdf:resource="&arith;#max"/>
        <drs:term_args rdf:parseType="Collection">
         <xsd:Integer rdf:datatype="&xsd;#Integer">0</xsd:Integer>
         <pr:valueOf>
             <pr:theParam rdf:resource="#O21"/>
             <pr:fromProcess rdf:resource="#S2"/>
          </pr:valueOf>
         </drs:term_args>
        </drs:Functional_term>
       </drs:term_args>
      </drs:Functional_term>
     </pr:valueFunction>
    </pr:OutputBinding>
   </pr:producedBinding>
  </pr:Produce>
 </pr:Sequence>
</pr:composedOf>
</pr:CompositeProcess>
```

Grounding a service

- Specifies the details of how to access the service
- A grounding be thought of as a *mapping* from an *abstract* to a *concrete* specification
- Both the *ServiceProfile* and the *ServiceModel* are thought of as abstract representations; only the *ServiceGrounding* deals with the concrete level of specification
- WSDL is chosen as a particular specification language
- OWL-S' concept of grounding is generally consistent with WSDL's concept of *binding*

OWL-S and WSDL



From: **The OWL Services Coalition**

OWL-S and WSDL

- An OWL-S atomic process corresponds to a WSDL *operation*
- The set of inputs and the set of outputs of an OWL-S atomic process each correspond to WSDL's concept of *message*.
- The types (OWL classes) of the inputs and outputs of an OWL-S atomic process correspond to WSDL's notion of *type*

From: **The OWL Services Coalition**

Grounding OWL-S with WSDL and SOAP

- Grounding OWL-S with WSDL and SOAP involves the construction of a WSDL service description with all the usual parts
- OWL class can either be defined within the WSDL *types* section, or defined in a separate document and referred to from within the WSDL description, using *owl-s-parameter*
- OWL-S extensions are introduced as follows:
  - In a *part* of the WSDL *message* definition, the *owl-s-parameter* attribute may be used to indicate the fully qualified name of the OWL-S input or output object (instance of class Parameter), to which this part of the message corresponds.
  - For those cases in which a message part uses an OWL type, the *encodingStyle* attribute, within the WSDL *binding* element, can be given a value such as ``**`http://www.w3.org/2002/07/owl`**"
  - In each WSDL *operation* element, the *owl-s-process* attribute may be used to indicate the name of the OWL-S atomic process, to which the operation corresponds.

From: **The OWL Services Coalition**

OWL-S/WSDL Document Example

```
<rdf:RDF xmlns:rdf= “http://www.w3.org/1999/02/22-rdf-syntax-ns”
    xmlns:rdfs= "http://www.w3.org/2000/01/rdf-schema“
    xmlns:owl = "http://www.w3.org/2002/07/owl”
    xmlns:xsd= "http://www.w3.org/2001/XMLSchema”
    xmlns:process= http://www.daml.org/services/owl-s/1.0/Process.owl”
    xmlns:grounding=
          "http://www.daml.org/services/owl-s/1.0/Grounding.owl"
    xmlns= “http://example.com/congo/CongoBuy.owl “>
<owl:Class rdf:ID="SignInData">
 . . .
</owl:Class>
<!-- The CongoBuy atomic process -->
<pr:AtomicProcess rdf:ID="CongoBuy">
  <pr:hasInput>
    <pr:Input ref:ID="In-BookName">
      <pr:parameterType rdf:about="&xsd;#string">
    </pr:Input>
  </pr:hasInput>
  <pr:hasInput>
    <pr:Input ref:ID="In-SignInData">
      <pr:parameterType rdf:resource="#SignInData">
    </pr:Input>
  </pr:hasInput>
  <pr:hasOutput>
    <pr:Output ref:ID="Out-Confirmation">
      <pr:parameterType rdf:resource="&xsd;#string">
    </pr:Output>
  </pr:hasOutput>
</pr:AtomicProcess>
```

OWL-S/WSDL Document Example

```
<?xml version="1.0"?>
<definitions name="CongoBuy“ targetNamespace="http://example.com/congo/congobuy.wsdl"
          xmlns:tns="http://example.com/congo/congobuy.wsdl"
          xmlns:congoOwl="http://example.com/congo/CongoBuy.owl#"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:owl-s-wsdl="http://www.daml.org/services/owl-s/wsdl/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">
 <message name="CongoBuyInput">
   <part name="BookName" owl-s-wsdl:owl-s-parameter="congoOwl:In-BookName"/>
   <part name="SignInInfo" owl-s-wsdl:owl-s-parameter="congoOwl:In-SignInInfo"/>
 </message>
 <message name="CongoBuyOutput">
   <part name="Confirmation" owl-s-wsdl:owl-s-parameter="congoOwl:Out-Confirmation"/>
 </message>
 <portType name="CongoBuyPortType">
   <operation name="BuyBook" owl-s-wsdl:owl-s-process="congoOwl:CongoBuy">
     <input message="tns:CongoBuyInput"/>
     <output message="tns:CongoBuyOutput"/>
   </operation>
 </portType>
 <binding name="CongoBuySoapBinding" type="tns:CongoBuyPortType">
   <soap:binding style=”rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="BuyBook">
     <soap:operation soapAction="http://example.com/congo/CongoBuy.owl#BuyBook"/>
      <input>
       <soap:body parts="BookName SignInInfo" use="encoded“
                  namespace="http://example.com/congo/“
                  encodingStyle="http://www.w3.org/2002/07/owl/"/>
      </input>
      <output>
       <soap:body parts="Confirmation" use="encoded"
                  namespace="http://example.com/congo/"
                  encodingStyle="http://www.w3.org/2002/07/owl/"/>
      </output>
    </operation>
  </binding>
  <service name="CongoBuyService">
    <documentation>My first OWL-S/WSDL service</documentation>
    <port name="CongoBuyPort" binding="tns:CongoBuySoapBinding">
      <soap:address location="http://example.com/congo/"/>
    </port>
  </service>
</definitions>
```

OWL-S' *Grounding* Class

- The OWL-S **`WsdlGrounding`** class is a mechanism by which the relevant WSDL constructs may be referenced in OWL-S by the following properties:
  - **`wsdlVersion:`** indicates the version of WSDL in use.
  - **`wsdlDocument:`** a WSDL document to which this grounding refers.
  - **`wsdlOperation:`** the WSDL operation corresponding to the given atomic process.
  - **`wsdlInputMessage:`** An object containing the URI of the WSDL message definition that carries the inputs of the given atomic process.
  - **`wsdlInputs:`** An object containing a list of mapping pairs, one for each message part of the WSDL input message.
  - **`wsdlOutputMessage:`** Similar to *wsdlInputMessage*, but for outputs.
  - **`wsdlOutputs`**: Similar to *wsdlInputs*, but for outputs.

OWL-S/WSDL Document Example

```
<!-- OWL-S Grounding Instance -->
<gr:WsdlGrounding rdf:ID="FullCongoBuyGrounding">
 <gr:hasAtomicProcessGrounding rdf:resource="#CongoBuyGrounding"/>
</gr:WsdlGrounding>
<gr:WsdlAtomicProcessGrounding rdf:ID="CongoBuyGrounding">
  <gr:owlsProcess rdf:resource="#CongoBuy">
   <gr:wsdlOperation>
    <gr:WsdlOperationRef>
      <gr:portType>
        <xsd:uriReference
          rdf:value="http://example.com/congo/congobuy.wsdl#CongoBuyPortType"/>
      </gr:portType>
      <gr:operation>
        <xsd:uriReference rdf:value="http://example.com/congo/congobuy.wsdl#BuyBook"/>
      </gr:operation>
    </gr:WsdlOperationRef>
  </gr:wsdlOperation>
  <gr:wsdlInputMessage
        rdf:resource="http://example.com/congo/congobuy.wsdl#CongoBuyInput"/>
  <gr:wsdlInputs rdf:parseType="owl:collection">
    <gr:wsdlInputMessageMap>
      <gr:owlsParameter rdf:resource="#In-BookName">
      <gr:wsdlMessagePart>
        <xsd:uriReference rdf:value="http://example.com/congo/congobuy.wsdl#BookName">
      </gr:wsdlMessagePart>
    </gr:wsdlInputMessageMap>
    <gr:wsdlInputMessageMap>
      <gr:owlsParameter rdf:resource="#In-SignInInfo">
      <gr:wsdlMessagePart>
        <xsd:uriReference rdf:value="http://example.com/congo/congobuy.wsdl#SignInInfo">
      </gr:wsdlMessagePart>
    </gr:wsdlInputMessageMap>
  </gr:wsdlInputs>
  <gr:wsdlOutputMessage
        rdf:resource="http://example.com/congo/congobuy.wsdl#CongoBuyOutput"/>
  <gr:wsdlOutputs rdf:parseType="owl:collection">
    <gr:wsdlOutputMessageMap>
      <gr:owlsParameter rdf:resource="#Out-Confirmation">
      <gr:wsdlMessagePart>
        <xsd:uriReference rdf:value="http://example.com/congo/congobuy.wsdl#Confirmation">
      </gr:wsdlMessagePart>
    </gr:wsdlOutputMessageMap>
  </gr:wsdlOutputs>
  <gr:wsdlVersion rdf:resource="http://www.w3.org/TR/2001/NOTE-wsdl-20010315">
  <gr:wsdlDocument>"http://example.com/congo/congobuy.wsdl"
  </gr:wsdlDocument>
</grounding:WsdlGrounding>
```

Chapters in the text-book that are less relevant to the exam

- Chapter 5 – Implementing Web services with Apache Axis
- Chapter 7 – Web services and J2EEE
- Chapter 10 – Web services reliable messaging
- Chapter 13 – Web services interoperability
- Chapter 14 – Web Services pragmatics