Semantic Web Services
Outline

- Semantic Web Services Basics
- Challenges in Web Services
- Semantics in Web Services
- Web Service Modeling Ontology
Outline

- Semantic Web Services Basics
- Challenges in Web Services
- Semantics in Web Services
- Web Service Modeling Ontology
The Vision of Semantic Web Services

- **Dynamic**
  - Web Services
    - UDDI, WSDL, SOAP
  - Semantic Web Services

- **Static**
  - WWW
    - URI, HTML, HTTP
  - Semantic Web
    - RDF(S), OWL, Rules

- **Syntactic**
- **Semantic**
The World Wide Web

- Largest document repository ever (> 25 billion Web pages indexed by Google)
- Highly distributed
  - Millions of publishers
  - No control over consistency of published content
- Web Technologies
  - HTTP for transferring documents
  - HTML for marking up documents
  - URI for addressing documents
- Most content on the Web is in natural language (HTML)
  - Natural language not suitable for machine reading
  - Current Web is “syntactic"
  - Problems in automatically:
    - Retrieving documents
    - Extracting relevant information from retrieved documents
    - Combining information from different sources
The Semantic Web

- Making the Web machine-readable.
- Publishing data in machine-readable format.
- Relating data on the Web to established vocabularies (ontologies).
- Ontologies specified in formal language to allow reasoning.
- Ontologies enable automation in:
  - Retrieval of relevant information.
  - Extracting relevant information from retrieved document.
  - Combination of information from different sources (as long as they are related to the same ontology).
Web Services

- Next step in software engineering:
  - 1960s: Procedural languages
  - 1980s: Object Orientation
  - 1990s: Component Orientation
  - 2000s: Web Services

- Loosely coupled, reusable components.

- Add new level of functionality to the Web.

- Web Service Technologies
  - SOAP for accessing Web Services
  - WSDL for describing Web Services
  - UDDI for publishing and looking up Web Services
Web Services Are Not Enough

- Like the current Web, Web Services are "syntactic".

- No automation in:
  - Finding services
  - Selecting services
  - Negotiation with service provider
  - Composing services
  - Executing services
Combining Semantic Web and Web Services

- **Semantic Web + Web Services = Semantic Web Services**

- Using Semantic Web technologies to describe Web Services.

- Enable automation in:
  - Publication
  - Discovery
  - Selection
  - Composition
  - Execution
Current Web Service Technologies

- **UDDI**
  - Registry of services

- **WSDL**
  - Service description

- **SOAP**
  - Message format
Service Oriented Architecture

Service Provider

- Publish: SOAP
- Bind: SOAP
- Find: SOAP

Service Broker

- WSDL

Service User

UDDI

Only Syntax, Agents can not understand meanings
UDDI

- **UDDI**: Universal, Description, Discovery, and Integration

- UDDI provides a mechanism for clients to find web services.

- A UDDI registry is similar to a CORBA trader, or it can be thought of as a DNS service for business applications.

- **UDDI consists of:**
  - White pages: Who is the service provider?
  - Yellow pages: What is the service providing?
  - Green pages: How the service can be used?

- **Public UDDIs**
  - Microsoft, IBM, others
  - Multitude of services is registered
  - Most services **no longer available**
  - Not used in practice

- **Private UDDIs**
WSDL

- **WSDL**: Web Services Description Language

An XML-based interface description language that is used for describing the functionality offered by a web service.

The terms used in WSDL are:

- **Service**: a set of system functions.

- **Port (Endpoint)**: defines the address or connection point to a web service and typically represented by a simple HTTP URL string.

- **PortType (Interface)**: Defines a web service by the collection of operations and the messages that are used to perform the operation.

- **Operation**: defines the actions
  - Name
  - Input/output
SOAP

- **SOAP**: Simple Object Access Protocol

- SOAP is basically a technology to allow for “RPC over the web”.

- SOAP is a message layout specification that defines a uniform way of passing XML-encoded data.

- Binding to HTTP as communication protocol.

- **SOAP Message**
  - **Envelope**
    - Sender address
    - Receiver address
    - Intermediate nodes
    - Security information
    - Format of the body
  - **Body**
    - Message content, typically XML
Other Web Service Standards

- **WS-Addressing**
  - Defining endpoints of services

- **WS-Security**
  - Comprehensive security framework for web services

- **WS-Policy**
  - Specification of policies for access, pricing, etc...

- etc...
Current Use of Web Services

- **On the Web**
  - **Amazon**
    - Finding products
    - Adding, deleting, updating shopping cart
    - Payment still over the Web
  - **Google**
    - Search Google via Web Service API
    - Accessing other Google services
    - Weather, stock tickers, etc...
  - Most public services use HTTP/GET, not SOAP

- **In Enterprises**
  - Many companies implementing Web Services
  - Example: Verizon
    - Before WSs: information about telephone numbers stored in 50 places
    - With WSs: one Web Service to retrieve telephone numbers
    - 1500 Business functions implemented using Web Services
    - Custom registry with services
Outline

- Semantic Web Services Basics
- **Challenges in Web Services**
- Semantics in Web Services
- Web Service Modeling Ontology
Advertising Services

- Current repositories (UDDI)
  - “polluted“: many registered services are not working now!

- Advertise functionality of the service.

- No non-functional aspects like:
  - Security
  - Accessibility
  - Cost
Discovering Services

- Finding services which provide **required functionality**.
- Requester needs to **specify** requirements.
- Mechanism for matching **required** with **advertised** functionality.
- Currently no way of specifying requirements.
- Discovery currently **manual**.
Selecting Services

- Given a set of services which meet requirements, select service with
  - best availability,
  - lowest cost,
  - best security measures,
  - etc..

- Select best service wrt. set of preferences.
  - Specification of requester's preferences.
  - Specification of non-functional properties of service.

- Selection currently manual.
Service Composition

- Required functionality requires **multiple** services to be invoked in **some combination**
  - Sequential execution
  - Parallel execution
  - Loops

- Specifying requirements
  - Single requirement specification
    - Find composition which fulfills requirement
  - Workflow description
    - Discovery of services for each task

- Composition currently manual process specification (e.g. BPEL, WSFL).
Example of Service Composition (1)

- Just simple services, without composition.
Example of Service Composition (2)

- Service composition to fulfill the requirement.
Service Execution

- How to invoke a service?
- If requester and provider use same format, current technologies suffice.
- What if actual formats differ?
  - We need translation of message content, in this situation.
Outline

- Semantic Web Services Basics
- Challenges in Web Services
- **Semantics in Web Services**
- Web Service Modeling Ontology
Semantics in Web Services (1)

- **Description of**
  - Functionality of services
    - e.g., searching, selling books, booking flight
  - User requirements
    - e.g., finding, buying books, booking flight
- **Web Service policies**
  - Permissions
  - Cost
  - etc...
- **User preferences**
- **Data in messages**
Matching descriptions
- functionality of service vs. user requirements
- policies vs. user preferences

Data translation

Web Service Composition
- Finding compositions
- Matching composition with user requirements
Functional Description of Services

- Describe functionality of Services
- Similar to software specification
- View web service as a function: $\textit{input} \rightarrow \textit{output}$
Rules for Policy Description

- Policies are often in the form of rules.
  - e.g., “if user is recognized business partner, then give discount“.
  - “if used encryption method is at least 3DES, then credit card detail may be transferred“.

- Use Semantic Web rules language for specifying policies.

- Initiatives for using rules to specify web service policies (e.g., policy RuleML).
Semantic Web Languages for Data Description

- Transfer data in RDF or OWL format
  - Advantage: More flexible than XML
  - Advantage: using ontology for describing structure of data

- Use ontologies for describing structure of data.

- If both parties use same ontology, no translation necessary.

- Data transformation based on ontology mapping (rules).
Outline

- Semantic Web Services Basics
- Challenges in Web Services
- Semantics in Web Services
- Web Service Modeling Ontology
Ontologies for Semantic Web Services

- Use ontologies to describe Web Services
  - DAML-S (since 2001)
  - OWL-S (since 2003)
    - Based on OWL-DL
  - WSMO (since 2004)
  - SWSO (since 2005)
    - Based on FLOWS
      (First-order Logic Ontology for Web Services)
The ontology of services provides three essential types of knowledge about a service:

- service profile,
- service model,
- service grounding.

The class **Service** provides an organizational point of reference for a declared Web Service.

1. The class Service “**presents**” a ServiceProfile: “What does the service provide for and require of agents?”
2. The class Service is “**describedBy**” a ServiceModel: “How does it work?”
3. The class Service “**supports**” a ServiceGrounding: “How to access the service?”
OWL-S Ontology

Service

- presents (what it does)
- supports (how to access it)
- describedby (how it works)

ServiceProfile

ServiceGrounding

ServiceModel
OWL-S — Service Profile (1)

- **Service Profile** specifies the functionality of the service and the conditions that must be satisfied for a successful result.

- The **profile** specifies what conditions result from the service, including the expected and unexpected results of the service activity.

- The **profile** represents two aspects of the functionality of the service:
  - the information transformation (represented by inputs and outputs)
  - the state change produced by the execution of the service (represented by preconditions and effects)
TABLE 11-1. ServiceProfile Property Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>Name of the service</td>
</tr>
<tr>
<td>textDescription</td>
<td>Provides a brief description of the service including what the service requires to work</td>
</tr>
<tr>
<td>contactInformation</td>
<td>Referring to individuals responsible for the service</td>
</tr>
</tbody>
</table>

TABLE 11-2. Service Profile Functional Description

Function Description: The Profile ontology defines the following properties of the Profile class for pointing to IOPEs:

- **hasParameter**: Ranges over a Parameter instance of the Process ontology. Parameter class models intuition that Inputs and Outputs (which are kinds of Parameters) are both involved in information transformation and therefore they are different from Preconditions and Effects.
- **hasInput**: Ranges over instances of Inputs as defined in the Process Ontology.
- **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 results of the service, as defined by the Result class in the Process ontology. It specifies under what conditions the outputs are generated.
OWL-S — Service Profile Ontology
OWL-S — Service Model

- Modeling services as processes.

- To give a detailed perspective on how to interact with a service, it can be viewed as a **process**.

- **Process** is a subclass of **ServiceModel**.

- A **process** is not a program to be executed. It is a specification of the ways a client may interact with a service.
OWL-S — Service Model
Process Ontology can have any number of inputs and outputs representing the information required for execution.

The OWL-S defines three types of processes:
- Atomic (single-step, directly invokable),
- Simple (single-step, but not directly invokable),
- Composite (decomposable into other processes).

Figure 11-3. Process ontology.

```xml
<owl:Class rdf:ID="Process">
  <rdfs:comment>The most general class of processes</rdfs:comment>
  <owl:disjointUnionOf rdf:parseType="owl:collection">
    <owl:Class rdf:about="#AtomicProcess"/>
    <owl:Class rdf:about="#SimpleProcess"/>
    <owl:Class rdf:about="#CompositeProcess"/>
  </owl:disjointUnionOf>
</owl:Class>
```
Atomic Process

- The atomic processes can be invoked directly and are executed in a single step.
- For each atomic process, there is a grounding that enables a service requester to construct messages.
- An AtomicProcess is a subclass of a Process

```xml
<owl:Class rdf:ID="AtomicProcess">
   <owl:subClassOf rdf:resource="#Process"/>
</owl:Class>
```
Simple Process

- Simple processes are not associated with a grounding. They are single-step executions.

- Simple processes are used as elements of abstraction:

- a simple process may be used either to provide a view of some atomic process, or a simplified representation of some composite process.

```xml
<owl:Class rdf:ID="SimpleProcess">
  <owl:subClassOf rdf:resource="#Process"/>
</owl:Class>

<rdfs:Property rdf:ID="realizedBy">
  <rdfs:domain rdf:resource="#SimpleProcess"/>
  <rdfs:range rdf:resource="#AtomicProcess"/>
  <owl:inverseOf rdf:resource="#realizes"/>
</rdfs:Property>

<rdfs:Property rdf:ID="expandsTo">
  <rdfs:domain rdf:resource="#SimpleProcess"/>
  <rdfs:range rdf:resource="#CompositeProcess"/>
  <owl:inverseOf rdf:resource="#collapsesTo"/>
</rdfs:Property>
```
Composite Process (1)

- Composite processes are decomposable into other processes. The decomposition shows how the various inputs and outputs are accepted.

```
<owl:Class rdf:ID="CompositeProcess">
  <owl:intersectionOf rdf:parseType="owl:collection">
    <owl:Class rdf:about="#Process"/>
    <owl:Restriction owl:cardinality="1">
      <owl:onProperty rdf:resource="#composedOf"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

- A CompositeProcess must have a composedOf property by which is indicated the control structure of the composite, using a ControlConstruct.

```
<rdf:Property rdf:ID="composedOf">
  <rdfs:domain rdf:resource="#CompositeProcess"/>
  <rdfs:range rdf:resource="#ControlConstruct"/>
</rdf:Property>
<owl:Class rdf:ID="ControlConstruct">
</owl:Class>
```
### TABLE 11-3. Process Constructs

<table>
<thead>
<tr>
<th>Construct</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>A list of Processes to be done in order.</td>
</tr>
<tr>
<td>Split</td>
<td>A bag of process components to be executed concurrently. Similar to other ontologies’ use of Fork, Concurrent, or Parallel.</td>
</tr>
<tr>
<td>Split + Join</td>
<td>Invoke elements of a bag of processes and synchronize.</td>
</tr>
<tr>
<td>Concurrent</td>
<td>Execute elements of a bag of processes concurrently.</td>
</tr>
<tr>
<td>Unordered</td>
<td>Allows the process components (specified as a bag) to be executed in some unspecified order, or concurrently.</td>
</tr>
<tr>
<td>Choice</td>
<td>Choose between alternatives and execute one.</td>
</tr>
<tr>
<td>If-Then-Else</td>
<td>If specified condition holds, execute “Then,” else execute “Else.”</td>
</tr>
<tr>
<td>Repeat-Until</td>
<td>Iterate execution of a bag of processes until a condition holds.</td>
</tr>
<tr>
<td>Repeat-While</td>
<td>Iterate execution of a bag of processes while a condition holds.</td>
</tr>
</tbody>
</table>
OWL-S — Grounding

- Specifies the details of how to access the service
  - Protocol and message format
  - Serialization, transport and addressing

- A *mapping* from an *abstract* to a *concrete* specification of those service description elements that are required for interacting with the service; i.e., the inputs and outputs of atomic processes.

- both the *ServiceProfile* and the *ServiceModel* are thought of as abstract representations; only the *ServiceGrounding* deals with the concrete level of specification.
WSMO

- **WSMO:** Web Service Modeling Ontology

- An ontology for Semantic Web Services.

- Provides conceptual model for SWS.

- Provide terminology for data exchanged between service requesters and providers.

- **Principles of WSMO:**
  - Ontology-based descriptions
  - Strict decoupling of components
  - Mediation between components
  - Interface, not implementation
Web Service Descriptions (1)

- Functionality offered by the Web Service.

- **Functional description**, in the form of a capability:
  - Assumptions
    - Cannot be checked
    - Usually indicate dependency on real world
  - Preconditions
    - Conditions over the input
  - Effects
    - Changes in the real world as a result of execution of the Web Service
  - Postconditions
    - Relation between the input and the output
Web Service Descriptions (2)

- **Behavioral description**, in the form of an *interface*:
  - Choreography
    - How to interact with the service
  - Orchestration
    - Use of external Web Service to realize the functionality
  - Both choreography and orchestration are decompositions of the capability.
Web Service Descriptions (3)
Goals Description

- Functionality requested from the Web Service
- Description symmetric to Web Service description:
  - Capability
  - Interface
WSMO Process

WSMO description

- describe goal
- discover services

WSMO Execution

- requester
- provider

describe service

advertise service

select service

inter-operate

inter-operate
Any Question...

amini@sharif.edu