Semantic Web Service Tutorial

HICSS 39

Katia Sycara
Massimo Paolucci

Michael Stollberg
Matthew Moran
Michal Zaremba
Mick Kerrigan
Emilia Cimpian

Stefania Galizia
Barry Norton
Liliana Cabral
John Domingue
Agenda

• Part I: Introduction to Semantic Web Services 09.00 – 09.30

• Part II: SWS Description Frameworks 09.30 – 12.00
  – OWL-S
  – WSMO
  coffee break 10.15 – 10.45
  lunch 12.00 – 01.00

• Part III: SWS Techniques and Systems 01.00 – 01.45
  – Discovery, Composition, Invocation, Mediation
  – OWL-S IDE, WSMX, IRS

• Part IV: Hands-On Session 01.45 – 04.00
  – Tools presentation
  coffee break 02.15 – 02.45
  – OWL-S IDE, WSMX
PART I: Introduction to Semantic Web Services

Michael Stollberg
Contents

• The vision of the Semantic Web

• Ontologies as the basic building block

• Current Web Service Technologies

• Vision and Challenges for Semantic Web Services
The Vision

- 500 million users
- more than 3 billion pages

Static

WWW
URI, HTML, HTTP
The Vision

Serious Problems in information

- finding
- extraction
- representation
- interpretation
- maintenance

Static

WWW
URI, HTML, HTTP

Semantic Web
RDF, RDF(S), OWL
The Vision

Dynamic Web Services
- UDDI, WSDL, SOAP

Static
- WWW
- URI, HTML, HTTP

Semantic Web
- RDF, RDF(S), OWL

brings the computer back as a device for computation
The Vision

bringing the web to its full potential

Dynamic

Web Services
UDDI, WSDL, SOAP

Static

WWW
URI, HTML, HTTP

Semantic Web Services
RDF, RDF(S), OWL
The Semantic Web

- the next generation of the WWW
- information has machine-processable and machine-understandable semantics
- not a separate Web but an augmentation of the current one
- ontologies as base technology
Ontology Definition

- unambiguous terminology definitions
- conceptual model of a domain (ontological theory)
- formal, explicit specification of a shared conceptualization
- machine-readability with computational semantics
- commonly accepted understanding
Ontology Example

**Concept**
conceptual entity of the domain

**Property**
attribute describing a concept

**Relation**
relationship between concepts or properties

**Axiom**
coherency description between Concepts / Properties / Relations via logical expressions

```
holds(Professor, Lecture) =>
Lecture.topic = Professor.researchField
```
Ontology Technology

To make the Semantic Web working we need:

• **Ontology Languages:**
  – expressivity
  – reasoning support
  – web compliance

• **Ontology Reasoning:**
  – large scale knowledge handling
  – fault-tolerant
  – stable & scalable inference machines

• **Ontology Management Techniques:**
  – editing and browsing
  – storage and retrieval
  – versioning and evolution Support

• **Ontology Integration Techniques:**
  – ontology mapping, alignment, merging
  – semantic interoperability determination
Web Services

• loosely coupled, reusable components
• encapsulate discrete functionality
• distributed
• programmatically accessible over standard internet protocols
• add new level of functionality on top of the current web

=> base technology for service oriented architectures (SOA) on the Web
The Promise of Web Services

web-based SOA as new system design paradigm

UDDI Registry

Points to Description

WSDL

Points to Service

Service Consumer

Finds Service

SOAP

Communicates with XML Messages

Web Service

Describes Service
WSDL

- Web Service Description Language
- W3C effort, WSDL 2 final specification phase

describes interface for consuming a Web Service:
- Interface: operations (in- & output)
- Access (protocol binding)
- Endpoint (location of service)
UDDI

- Universal Description, Discovery, and Integration Protocol
- OASIS driven standardization effort

Registry for Web Services:
- provider
- service information
- technical access
SOAP

- Simple Object Access Protocol
- W3C Recommendation

XML data transport:
- sender / receiver
- protocol binding
- communication aspects
- content
Lackings of WS Technology

• current technologies allow usage of Web Services
• but:
  – only syntactical information descriptions
  – syntactic support for discovery, composition and execution
  => *Web Service usability, usage, and integration needs to be inspected manually*
  – no semantically marked up content / services
  – no support for the Semantic Web

=> current Web Service Technology Stack failed to realize the promise of Web Services
Semantic Web Services

Semantic Web Technology

• allow machine supported data interpretation
• ontologies as data model

+ 

Web Service Technology

automated discovery, selection, composition,
and web-based execution of services

=> Semantic Web Services as integrated solution for realizing the vision of the next generation of the Web
Semantic Web Services

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

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

• define semantically driven technologies for automation of the Web Service usage process *(Web Service aspect)*
Web Service Usage Process

1. Deployment  create & publish Web service description
2. Discovery  determine usable services for a request
3. Composition  combine services to achieve a goal
4. Selection  choose most appropriate service among the available ones
5. Mediation  solve mismatches (data, protocol, process) that hamper interoperation
6. Execution  invoke Web services following programmatic conventions
Web Service Execution Support

- **Monitoring**: control the execution process
- **Compensation**: provide transactional support and undo or mitigate unwanted effects
- **Replacement**: facilitate the substitution of services by equivalent ones
- **Auditing**: verify that service execution occurred in the expected way
PART II: Semantic Web Service Ontologies

Katia Sycara
Michael Stollberg
Contents

• **OWL-S**
  - Upper Ontology
  - Service Profile
  - Process Model
  - Service Grounding

• **WSMO**
  - WSMO top level notions
  - Choreography and Orchestration
  - Mediation

• **Differences and Commonalities**
OWL-S

Katia Sycara
OWL-S Ontology

- OWL-S is an OWL ontology to describe Web services
- OWL-S leverages on OWL to
  - Support capability based discovery of Web services
  - Support automatic composition of Web Services
  - Support automatic invocation of Web services

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

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

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

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

Service Profile
- Presented by a service.
- Represents what the service provides
- Two main uses:
  1. Advertisements of Web Services capabilities
  2. Request of Web services with a given set of capabilities
OWL-S Profile in a Nutshell

• Describes Web service
  – What capabilities it provides:
    • What transformation the service computes
    • Type of service and products
  – General features such as
    • Agent providing the service
    • Security requirements
    • Quality guarantees of service

• Primary role: to assist discovery
  – Allows capability based search
  – Allows selection based on requirements of the requester

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

Capability Description

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

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

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

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

- **Service type**
  - What kind of service is provided (eg selling vs distribution)

- **Product**
  - Product associated with the service (eg travel vs books vs auto parts)
OWL-S Service Profile
Additional Properties

• **Security Parameters**
  – Specify the security capabilities of a Web service (eg support X509 Encryption)
  – Specify the security requirements of a Web service (eg a client should be able to provide X509 Encryption)

• **Quality rating**
  – What level of service quality does the Web service provide?

• **Description with standard business taxonomies**
  – How would the service be classified in standard taxonomies such as UNSPSC or NAICS?

This is not a closed set, new properties can be added using existing ontologies
Process Model

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

Facilitates
- Web service invocation
- Composition of Web services
- Monitoring of interaction
Viewpoints of Process Model

- Three viewpoints of a Web service
  - **Glass Box:**
    - The Web service reveals all its internal structure
    - Which parts of the service it performs in-house, which one it subcontracts, etc
  - **Black Box:**
    - The Web service model does not reveal anything about the internal working of the service
    - It just specifies what data it gathers and what data it sends back
  - **Grey Box:**
    - The Web service selectively hides some parts of its Process Model, while it publicizes others
Definition of Process

• A Process represents a transformation (function). It is characterized by four parameters
  – **Inputs**: the inputs that the process requires
  – **Preconditions**: the conditions that are required for the process to run correctly
  – **Outputs**: the information that results from (and is returned from) the execution of the process
  – **Results**: a process may have different outcomes depending on some condition
    • **Condition**: under what condition the result occurs
    • **Constraints on Outputs**
    • **Effects**: real world changes resulting from the execution of the process
Motivation for Results

• Processes may terminate in exceptional states:
  – The credit company may fail to charge the credit card
  – The book may be out of stock
  – The deliver of the goods may fail

• Results support modeling of non-deterministic outcomes of Web services
  – The condition specifies when an outcome is generated
  – Each outcome is characterized by
    • a set of constraints on outputs
    • a set of effects
Example of Process

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

**Process**
- **Atomic**
  - Invokable bound to grounding
- **Simple**
  - Provides abstraction, encapsulation etc.
- **Composite**
  - Defines a workflow composed of process performs
Process Model Organization

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

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

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

• Composite Processes specify how processes work together to compute a complex function
• Composite processes define

  1. Control Flow
     Specify the temporal relations between the executions of the different sub-processes

  2. Data Flow
     Specify how the data produced by one process is transferred to another process
Example of Composite Process

- **Sequence** BookFlight

**Control Flow Links**
Specify order of execution

**Data-Flow Links**
Specify transfer of data

- **Perform**
  - Airline
  - Get Flights
  - Select Flight

**Perform statements**
Specify the execution of a process

- **Flights**
  - Depart
  - Arrive
**Perform Construct**

- *Perform* provides invocation mechanism
  - Specify context of process execution
    - input data flow
    - hooks for output data flow
- Distinction between definition and invocation of a process
  - Definition specifies the process’ I/P/R
  - Perform specify when the process is invoked and with what parameters
Control Flow

- Processes can be chained to form a workflow
- OWL-S supports the following control flow constructs
  - *Sequence/Any-Order*: represents a list of processes that are executed in sequence or arbitrary order
  - *Conditionals*: if-then-else statements
  - *Loops*: while and repeat-until statements
  - *Multithreading and synchronization*: split process in multiple threads, and rendezvous (joint) points
  - *Non-deterministic choices*: (arbitrarily) select one process of a set
Data Flow

Dataflow allows information that is transferred from process to process.

**Output → Input:**

The information produced by one process is transferred to another in the same control construct

**Input → Input:**

The information received by a composite process is transferred to the sub-processes

**Output → Output:**

The information produced by a subprocess is transferred to a super-process
Process Model: take home lesson

• Service Model describes
  – Set of processes that define the operations performed by the Web service
  – Control flow describing the temporal flow of processes
  – Data flow describing the transfer of information between sub-processes
Service Grounding

- Service Grounding
  - Provides a specification of service access information.
  - Service Model + Grounding give everything needed for using the service
  - Builds upon WSDL to define message structure and physical binding layer

- Specifies:
  - communication protocols, transport mechanisms, communication languages, etc.
Rationale of Service Grounding

• Provides a specification of service access information.

• Service Model + Grounding give everything needed for using the service
  – Service description is for reasoning about the service
    • Decide what information to send and what to expect
  – Service Grounding is for message passing
    • Generate outgoing messages, and get incoming messages
    • Mapping XML Schemata to OWL concepts

• Builds upon **WSDL** to define message structure and physical binding layer
Mapping OWL-S / WSDL 1.1

- **Operations** correspond to Atomic Processes
- **Input/Output** messages correspond to Inputs/Outputs of processes
Example of Grounding

Sequence
BookFlight

Perform
Get Flights

Perform
Select Flight

Get Flights Op

Select Flight op

Airline

Flight

WSDL
Result of using the Grounding

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

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

• Allows any web service to be represented using OWL-S
  – For example: Amazon.com
Handling stateful vs stateless Web services

1. Stateless Web services
   - The server does not maintain the state of the computation
   - Dataflow links specify how the client communicate the state to the service

2. Stateful Web services
   - The service does maintain the state
   - No need of dataflow links since transfer of information is opaque to the client
Representing Stateful Web services

**Stateless:** no information is transferred between the two operations
Representing Stateless Web services

Stateful: information is recorded by the server, no need of transfer between the two operations.
Conclusion  OWL-S section

• OWL-S provides a language for the description of Web services
  – Service Profile provides description of capabilities of Web Service
    • Allows capability-based discovery
  – Process Model provides the description of how to use a Web service
    • Allows automatic invocation of Web service
  – Service Grounding maps Atomic Processes into WSDL operations
    • Allows separation between description and implementation
    • Supports description of arbitrary Web services
Web Service Modeling Ontology
WSMO

Michael Stollberg
Outline

• WSMO Working Groups

• Top Level Notions
  – Ontologies
  – Web Services
  – Goals
  – Mediators
WSMO Working Groups

A Conceptual Model for SWS

A Formal Language for WSMO

A Rule Language for the Semantic Web

Execution Environment for WSMO
WSMO Top Level Notions

Objectives that a client wants to achieve by using Web Services

Formally specified terminology of the information used by all other components

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

Connectors between components with mediation facilities for handling heterogeneities

WSMO D2, version 1.2, 13 April 2005 (W3C submission)
Non-Functional Properties

*relevant, non-functional aspects for WSMO elements*

- Dublin Core Metadata Set:
  - complete item description
  - used for resource management
- Versioning Information
  - evolution support
- Quality of Service Information
  - availability, stability
- Other
  - owner, financial
## Non-Functional Properties List

<table>
<thead>
<tr>
<th>Dublin Core Metadata</th>
<th>Quality of Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contributor</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Coverage</td>
<td>NetworkRelatedQoS</td>
</tr>
<tr>
<td>Creator</td>
<td>Performance</td>
</tr>
<tr>
<td>Description</td>
<td>Reliability</td>
</tr>
<tr>
<td>Format</td>
<td>Robustness</td>
</tr>
<tr>
<td>Identifier</td>
<td>Scalability</td>
</tr>
<tr>
<td>Language</td>
<td>Security</td>
</tr>
<tr>
<td>Publisher</td>
<td>Transactional</td>
</tr>
<tr>
<td>Relation</td>
<td>Trust</td>
</tr>
<tr>
<td>Rights</td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td>Other</td>
</tr>
<tr>
<td>Subject</td>
<td>Financial</td>
</tr>
<tr>
<td>Title</td>
<td>Owner</td>
</tr>
<tr>
<td>Type</td>
<td>TypeOfMatch</td>
</tr>
<tr>
<td></td>
<td>Version</td>
</tr>
</tbody>
</table>
WSMO Ontologies

Objectives that a client wants to achieve by using Web Services

- Capability (functional)
- Interfaces (usage)

Formally specified terminology of the information used by all other components

Connectors between components with mediation facilities for handling heterogeneities
Ontology Usage & Principles

- **Ontologies are the ‘data model’ throughout WSMO**
  - all WSMO element descriptions rely on ontologies
  - all data interchanged in Web Service usage are ontologies
  - Semantic information processing & ontology reasoning

- **WSMO Ontology Language WSML**
  - conceptual syntax for describing WSMO elements
  - logical language for axiomatic expressions (WSML Layering)

- **WSMO Ontology Design**
  - **Modularization:** import / re-using ontologies, modular approach for ontology design
  - **De-Coupling:** heterogeneity handled by **OO Mediators**
Ontology Specification

- Non functional properties (see before)
- Imported Ontologies importing existing ontologies where no heterogeneities arise
- Used mediators OO Mediators (ontology import with terminology mismatch handling)

Ontology Elements:

- Concepts set of concepts that belong to the ontology, incl.
- Attributes set of attributes that belong to a concept
- Relations define interrelations between several concepts
- Functions special type of relation (unary range = return value)
- Instances set of instances that belong to the represented ontology
- Axioms axiomatic expressions in ontology (logical statement)
WSMO Web Services

Objectives that a client wants to achieve by using Web Services

- Formally specified terminology of the information used by all other components

Connectors between components with mediation facilities for handling heterogeneities

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

- complete item description
- quality aspects
- Web Service Management

Non-functional Properties
DC + QoS + Version + financial

Capability
functional description

- Advertising of Web Service
- Support for WS Discovery

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

realization of functionality by aggregating other Web Services
- functional decomposition
- WS composition

client-service interaction interface for consuming WS
- External Visible Behavior
- Communication Structure
- ‘Grounding’

Choreography --- Service Interfaces --- Orchestration
Capability Specification

- **Non functional properties**
- **Imported Ontologies**
- **Used mediators**
  - *OO Mediator*: importing ontologies with data level mismatch resolution
  - *WG Mediator*: link to a Goal wherefore service is not usable a priori
- **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
- **Post-conditions**
  describes the result of the Web Service in relation to the input, and conditions on it
- **Effects**
  conditions on the state of the world that hold after execution of the Web Service (i.e. changes in the state of the world)
Choreography & Orchestration

- **VTA example:**

  - **Choreography** = how to interact with the service to consume its functionality
  - **Orchestration** = how service functionality is achieved by aggregating other Web Services
Choreography Interfaces

*Interface for consuming Web Service*

- **External Visible Behavior**
  - those aspects of the workflow of a Web Service where Interaction is required
  - described by workflow constructs: sequence, split, loop, parallel
- **Communication Structure**
  - messages sent and received
  - their order (communicative behavior for service consumption)
- **Grounding**
  - executable communication technology for interaction
  - choreography related errors (e.g. input wrong, message timeout, etc.)
- **Formal Model**
  - reasoning on Web Service interfaces (service interoperability)
  - semantically enabled mediation on Web Service interfaces
Orchestration Aspects

Behavior for Interaction with aggregated Web Services

- decomposition of service functionality
- other Web services consumed via their choreography interfaces
WSMO Web Service Interfaces

- behavior interfaces of Web services and clients for “peer-2-peer” interaction
- Choreography and Orchestration as sub-concepts of Service Interface with common description language
- service interface description aspects:
  1. represent the dynamics of information interchange during service consumption and interaction
  2. support ontologies as the underlying data model
  3. appropriate communication technology for information interchange
  4. sound formal model / semantics of service interface specifications in order to allow advanced reasoning on them
  5. support higher-level process constructs for more complex reasoning tasks
  6. provide graphical representation for editing and maintenance
Semantic Web Services, HICSS 39, Kauai (Hawaii), 04 January 2006

Service Interface Description

User Language (UML2 Activity Diagrams)
graphical representation for choreography & orchestration descriptions

Downwards Translation
User Language -> Formal Model

Formal Model:
“ontologized ASMs” as sound formalism

semantic data model

(WSMO) Ontologies as data model:
- every resource description based on ontologies
- every data element interchanged is ontology instance

Grounding:
- making service interfaces executable
- currently grounding to WSDL

execution support
Ontologized Abstract State Machines

• Vocabulary $\Omega$:
  – ontology schema(s) used in service interface description
  – usage for information interchange: in, out, shared, controlled

• States $\omega(\Omega)$:
  – a stable status in the information space
  – defined by attribute values of ontology instances

• Guarded Transition $GT(\omega)$:
  – state transition
  – general structure: if (condition) then (update)
    • condition on current state, update = changes in state transition
    • all $GT(\omega)$ whose condition is fulfilled fire in parallel
WSMO Goals

Objectives that a client wants to achieve by using Web Services

Formally specified terminology of the information used by all other components

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

Connectors between components with mediation facilities for handling heterogeneities
Goals

Client Objective Specification along with all information needed for automated resolution

• **Goal-driven Approach**, derived from AI rational agent approach
  - ontological de-coupling of Requester and Provider
  - ‘intelligent’ mechanisms detect suitable services for solving the Goal
  - service re-use & knowledge-level client side support

• **Usage of Goals within Semantic Web Services**
  – A Requester (human or machine) defines a Goal to be resolved independently (i.e. subjectively) on the knowledge level
  – SWS techniques / systems automatically determine Web Services to be used for resolving the Goal (discovery, composition, execution, etc.)
  – Goal Resolution Management is realized in implementations
Goal-driven Architecture

Client-Side

Goal
- objective (desired final state)
- input for service usage
- goal resolution constraints, preferences, and policies

Goal Resolution Plan
- goal resolution algorithm
- decomposition (optional)
- service usage / invocation

defines

corresponds to / creation of

Service-Side

(Web) Service Implementation
- functional
- behavioral

(not of interest here)

domain knowledge

Ontology
Ontology

Domain Knowledge
Ontology
Ontology
Mediation

• Heterogeneity …
  – mismatches on structural / semantic / conceptual / level
  – occur between different components that shall interoperate
  – especially in distributed & open environments like the Internet

• Concept of Mediation (Wiederhold, 94):
  – *Mediators* as components that resolve mismatches
  – declarative approach:
    • semantic description of resources
    • ‘intelligent’ mechanisms that resolve mismatches independent of content
  – mediation cannot be fully automated (integration decision)

• Levels of Mediation within Semantic Web Services:
  (1) Data Level: heterogeneous Data Sources
  (2) Functional Level: heterogeneous Functionalities
  (3) Protocol & Process Level: heterogeneous Communication Processes
WSMO Mediators Overview

Legend:
- Technique used
- Imports / Reuses
- Correlation
Mediator Usage
OWL-S and WSMO

Commonalities and Differences
**OWL-S and WSMO**

- **OWL-S** = ontology and language to describe Web services
- **WSMO** = ontology and language for core elements of Semantic Web Service systems

**Main Description Elements Correlation:**

<table>
<thead>
<tr>
<th>OWL-S profile</th>
<th>≈ WSMO capability + non-functional properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWL-S Process Model</td>
<td>≈ WSMO Service Interfaces</td>
</tr>
<tr>
<td>OWL-S Grounding</td>
<td>≈ current WSMO Grounding</td>
</tr>
</tbody>
</table>
Mediation in OWL-S and WSMO

• OWL-S does not have an explicit notion of mediator
  – Mediation is a by-product of the orchestration process
    • E.g. protocol mismatches are resolved by constructing a plan that
      coordinates the activity of the Web services
  – …or it results from translation axioms that are available to the
    Web services
    • It is not the mission of OWL-S to generate these axioms

• WSMO regards mediators as key conceptual elements
  – Different kinds of mediators:
    • OO Mediators for ensuring semantic interoperability
    • GG, WG mediators to link Goals and Web Services
    • WW Mediators to establish service interoperability
  – Reusable mediators
  – Mediation techniques under development
Semantic Representation

• OWL-S and WSMO adopt a similar view on the need of ontologies and explicit semantics but they rely on different logics

  – OWL-S is based on OWL/SWRL
    • OWL represent taxonomical knowledge
    • SWRL provides inference rules
    • FLOWS as formal model for process model

  – WSMO is based on
    • WSML a family of languages with a common basis for compatibility and extensions in the direction of Description Logics and Logic Programming
    • Ontologizes Abstract State Machines and formal model for Service Interface Descriptions
OWL vs WSML

OWL Full

OWL DL

OWL Lite

WSML Full

WSML DL

WSML Core

WSML Rule

WSML Flight

full RDF(S) support

equals

subset

First Order Logic

Description Logics

Description Logics

Logic Programming

equals

subset
## Summary

### Current Web Service Technologies

<table>
<thead>
<tr>
<th></th>
<th>OWL-S</th>
<th>WSMO</th>
<th><strong>current Web Service technologies</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discovery</strong></td>
<td></td>
<td><strong>What it does</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>Goals and Web Services (capability)</td>
<td><strong>UDDI API</strong></td>
</tr>
<tr>
<td><strong>Consumption &amp; Interaction</strong></td>
<td>Process Model</td>
<td>Service Interfaces (Choreography + Orchestration)</td>
<td><strong>BPEL4WS</strong></td>
</tr>
<tr>
<td><strong>Invocation</strong></td>
<td>Grounding+ WSDL/SOAP</td>
<td>Grounding (WSDL / SOAP, ontology-based)</td>
<td><strong>WSDL/SOAP</strong></td>
</tr>
</tbody>
</table>

*How to invoke*
PART III:
Semantic Web Service Techniques and Systems

Michael Stollberg
Contents

• The “Virtual Travel Agency Example”
  – Goal and Web service description
  – discovery
  – mediation

• SWS tools and systems
  – Web Service Execution Environment WSMX
  – OWL-S Integrated Development Environment
  – Internet Reasoning Service IRS
SWS Challenges

- Web services as loosely coupled components that shall interoperate dynamically and automatically
- Techniques required for:
  - **Discovery**
    - how are Web services found and selected?
  - **Composition**
    - how to aggregate Web Services into a complex functionality?
  - **Conversation**
    - how to ensure automated interaction of Web Services?
  - **Invocation**
    - how to access and invoke Semantic Web Services?
  - **Mediation**
    - how are data and protocol mismatches resolved?
- Systems for automated Web service usage:
  - resource editing and management
  - functional components
  - APIs, execution control, integrated & flexible architectures
Virtual Travel Agency Use Case

- Michael is employed in DERI Austria and wants to book a flight and a hotel for the HICSS-39 conference
- the start-up company VTA provides tourism and business travel services based on Semantic Web Service technology

=> how does the interplay of Michael, VTA, and other Web Services look like?
Domain Ontologies

- All terminology used in resource descriptions are based on ontologies and all information interchanged should be ontology instances.

- Domain Ontologies needed for this Use Case:
  - Trip Reservation Ontology, Location Ontology, Date and Time Ontology, Purchase Ontology, … possibly more.

- Ontology Design for the Semantic Web
  - "real ontologies, no crappy data models" (Dieter Fensel)
  - (re-)use existing, widely accepted ontologies
  - Modular ontology design
  - … is a very difficult and challenging task
    - Determine agreed conceptualization of domain
    - Correct formalization (e.g. misuse of is_a / part_of relations)

=> Requires expertise in knowledge engineering.
Trip Reservation Ontology

- defines the terminology for trips (traveling, accommodation, holiday / business travel facilities) and reservations
- provided by community of interest (e.g. Austrian Tourism Association)

- main concepts:
  - TRIP
    - describes a trip (a journey between locations)
    - passenger, origin & destination, means of travel, etc.
  - RESERVATION
    - describes reservations for tickets, accommodation, or complete trips
    - customer, trip, price, payment
  - RESERVATION REQUEST / OFFER / CONFIRMATION

- uses other ontologies:
  - Location Ontology for origin & destination specification
  - Date and Time Ontology for departure, arrival, duration information
  - Purchase Ontology for payment related aspects
Goal Description

• “book flight and hotel for the HICSS-39 for Michael”
• goal capability postcondition: get a trip reservation for this

```wsmo
goal _"http://www.wsmo.org/examples/goals/hicss39"
importsOntology {"http://www.wsmo.org/ontologies/tripReservationOntology", …}
capability
    postcondition definedBy
        ?tripReservation memberOf tr#reservation[customer hasValue fof#michael,
            reservationItem hasValue ?tripHICSS] and
        ?tripHICSS memberOf tr#trip[
            passenger hasValue fof#michael,
            origin hasValue loc#innsbruck,
            destination hasValue loc#kauai,
            meansOfTransport hasValue ?flight,
            accomodation hasValue ?hotel] and
        ?flight[airline hasValue tr#staralliance] memberOf tr#flight and
        ?hotel[name hasValue “Grand Hyatt Kauai Resort”] memberOf tr#hotel .
```
VTA Service Description

- book tickets, hotels, amenities, etc.
- capability description (pre-state)

```
capability VTAcapability
sharedVariables {?creditCard, ?initialBalance, ?item, ?passenger}
precondition
  definedBy
  ?reservationRequest[
    reservationItem hasValue ?item,
    passenger hasValue ?passenger,
    payment hasValue ?creditcard,
  ] memberOf tr#reservationRequest and
  (?item memberOf tr#trip) or (?item memberOf tr#ticket)) and
  ?creditCard[balance hasValue ?initialBalance] memberOf po#creditCard .

assumption
  definedBy
  po#validCreditCard(?creditCard) and
  (?creditCard[type hasValue po#visa] or ?creditCard[type hasValue po#mastercard]).
```
VTA Service Description

- capability description (post-state)

```xml
postcondition
  definedBy
    ?reservation[
      reservationItem hasValue ?item,
      customer hasValue ?passenger,
      payment hasValue ?creditcard
    ] memberOf tr#reservation .

assumption
  definedBy
    reservationPrice(?reservation, ?tripPrice) and
    ?finalBalance= (?initialBalance - ?ticketPrice) and
    ?creditCard[po#balance hasValue ?finalBalance] .
```
Web Service Discovery

Objective: „book a flight and a hotel for me for the HICSS-39.“

Goal definition

Michael has

Service Registry searches

WS Discoverer result set includes

VTA
Discovery Techniques

• different techniques available
  – trade-off: ease-of-provision <-> accuracy
  – resource descriptions & matchmaking algorithms

Key Word Matching
  match natural language key words in resource descriptions

Controlled Vocabulary
  ontology-based key word matching

Semantic Matchmaking
  … what Semantic Web Services aim at
Matchmaking Notions & Intentions

Exact Match:
\[ G, WS, O, M \models \forall x. (G(x) \iff WS(x)) \]

PlugIn Match:
\[ G, WS, O, M \models \forall x. (G(x) \implies WS(x)) \]

Subsumption Match:
\[ G, WS, O, M \models \forall x. (G(x) \subseteq WS(x)) \]

Intersection Match:
\[ G, WS, O, M \models \exists x. (G(x) \land WS(x)) \]

Non Match:
\[ G, WS, O, M \models \neg \exists x. (G(x) \land WS(x)) \]

Discoverer Architecture

- Discovery as central Semantic Web Services technology
- Integrated Discoverer Architectures (under construction):

  - Resource Repository (UDDI or other)
  - Keyword-/Classification-based Filtering
  - Controlled Vocabulary Filtering
  - Semantic Matchmaking
  - Usable Web Service

  - Efficient narrowing of search space (relevant services to be inspected)

  - Retrieve Service Descriptions
  - Invoke Web Service
Choreography Discovery

Graphical representation of a choreography with interfaces and capabilities for booking a trip, including:

- VTA, providing a 'Trip Booking' capability with interfaces for getting requests, providing offers, receiving selections, and sending confirmations.
- Flight WS, offering a capability to book flights.
- Hotel WS, providing a capability to book hotels.

The diagram highlights:

- Both choreography interfaces given ("static")
- Correct & complete consumption of VTA

Question:

- VTA Orchestration & Choreography Interfaces of aggregated WS given

Conclusion:

- **Choreography Discovery** as a central reasoning task in Service Interfaces
- "choreographies" do not have to be described, only existence determination
Choreography Discovery

• a valid choreography exists if:
  1) **Signature Compatibility**
     • homogeneous ontologies
     • compatible in- and outputs
  2) **Behavior Compatibility**
     • start state for interaction
     • a termination state can be reached without any additional input
Behavior Compatibility Example

Goal Behavior Interface

\[ \Omega_G(\omega\emptyset) = \{\emptyset\} \]

\[ \text{if } \emptyset \text{ then request} \]

\[ \Omega_G(\omega_1) = \{\text{request(out)}\} \]

\[ \text{if cnd1(offer) then changeReq} \]

\[ \Omega_G(\omega_{2a}) = \{\text{offer(in)}, \text{changeReq(out)}\} \]

\[ \text{if cnd2(offer) then order} \]

\[ \Omega_G(\omega_{2b}) = \{\text{offer(in)}, \text{order(out)}\} \]

\[ \text{if conf then } \emptyset \]

\[ \Omega_G(\omega_3) = \{\text{offer(in)}, \text{conf(in)}\} \]

VTA Behavior Interface

\[ \Omega_{VTA}(\omega\emptyset) = \{\emptyset\} \]

\[ \text{if request then offer} \]

\[ \Omega_{VTA}(\omega_1) = \{\text{request(in)}, \text{offer(out)}\} \]

\[ \text{if changeReq then offer} \]

\[ \Omega_{VTA}(\omega_{2a}) = \{\text{changeReq(in)}, \text{offer(out)}\} \]

\[ \text{if order then conf} \]

\[ \Omega_{VTA}(\omega_{2b}) = \{\text{order(in)}, \text{conf(out)}\} \]

valid choreography existent
Orchestration Validation Example

VTA Web Service Orchestration

if Ø then (FWS, flightRequest)
if flightOffer then (HWS, hotelRequest)
if selection then (FWS, flightBookingOrder)
if selection, flightBookingConf then (HWS, hotelBookingOrder)

Fligh WS Behavior Interface

Start (VTA, FWS)
Termination (VTA, FWS)

Hotel WS Behavior Interface

Start (VTA, HWS)
Termination (VTA, HWS)

Orchestration is valid if valid choreography exists for interactions between the orchestrating and each aggregated Web Service, done by choreography discovery
Mediation

• Heterogeneity as inherent characteristic of (Semantic) Web:
  – heterogeneous terminology
  – heterogeneous languages / formalisms
  – heterogeneous communication protocols and business processes

• WSMO identifies Mediators as top level element, i.e. central aspect of Semantic Web Services
  – levels of mediation: data, protocol, processes
  – WSMO Mediator types

• Approach: declarative, generic mismatch resolution
  – classification of possible & resolvable mismatches
  – mediation definition language & mediation patterns
  – execution environment for mappings
Data Level (OO) Mediation

- Related Aspects / Techniques:
  - Ontology Integration (Mapping, Merging, Alignment)
  - Data Lifting & Lowering
  - Transformation between Languages / Formalisms

- Terminology Mismatch Classification
  - Conceptualization Mismatches
    - same domain concepts, but different conceptualization
    - different levels of abstraction
    - different ontological structure
    => resolution only incl. human intervention
  - Explication Mismatches
    - mismatches between:
      \( T \) (Term used) \( D \) (definition of concepts), \( C \) (real world concept)
    => automated resolution partially possible
Ontology Mapping Language

• Language Neutral Mapping Language
  – mapping definitions on meta-layer (i.e. on generic ontological constructs)
  – independent of ontology specification language
  – “Grounding” to specific languages for execution (WSML, OWL, F-Logic)

• Main Features:
  – Mapping Document (sources, mappings, mediation service)
  – direction of mapping (uni- / bidirectional)
  – conditions / logical expressions for data type mismatch handling,
    restriction of mapping validity, and complex mapping definitions
  – mapping constructs:
    • classMapping, attributeMapping, relationMapping (between similar constructs)
    • classAttributeMapping, classRelationMapping, classInstanceMapping
    • instanceMapping (explicit ontology instance transformation)
  – mapping operators:
    • =, <, <=, >, >=, and, or, not
    • inverse, symmetric, transitive, reflexive
    • join, split
Mapping Language Example

this allows to transform the instance ‘michael’ of concept person in ontology O2 into a valid instance of concept ‘adult’ in ontology O1
• if a choreography does not exist, then find an appropriate WW Mediator that:
  – resolves possible mismatches to establish signature compatibility (OO Mediator usage)
  – resolves process / protocol level mismatches in to establish behavior compatibility
Process Mediation – Addressed Mismatches
Process Mediation Example

REQUEST

- itinerary[origin, destination, date]
- time
- price

Processes Mediator

SERVICE

- origin
- destination
- itinerary[origin, destination]
- date
- itinerary [route, date, time, price]
Process Mediation Example

REQUEST

- itinerary [origin, destination, date]
- time
- price

Processes Mediator

SERVICE

- origin
- destination
- itinerary [origin, destination]
- date
- itinerary [route, date, time, price]
Process Mediation Example

REQUEST

- itinerary[origin, destination, date]
- time
- price

Processes Mediator

SERVICE

- origin
- destination
- itinerary[origin, destination]
- date
- itinerary [route, date, time, price]
Process Mediation Example

REQUEST

- itinerary [origin, destination, date]
- time
- price

Processes Mediator

SERVICE

- origin
- destination
- itinerary [origin, destination]
- date
- itinerary [route, date, time, price]
Process Mediation Example

Processes Mediator

REQUEST

- itinerary [origin, destination, date]
- time
- price

SERVICE

- origin
- destination
- itinerary [origin, destination]
- date
- itinerary [route, date, time, price]
SWS Tools and Systems

1. **OWL-S Integrated Development IDE**
   - OWL-S tool suite
   - WS implementation, deployment, discovery, invocation, and verification

2. **The Web Service Execution Environment WSMX**
   - integrated Semantic Web Service system
   - WSMO reference implementation

3. **Internet Reasoning Service IRS**
   - infrastructure for Semantic Web services
   - IRS server acts as broker, as well as publisher
   - IRS client allows goal-based invocation
OWL-S IDE (CMU)

Integration of WS implementation, deployment, discovery, invocation and verification
Integrated WS Development cycle

• OWL-S IDE aims at automating WS-Development and invocation cycle
  – Based on Eclipse to support WS programmers
  – (Semi) Automated generation of WSDL and OWL-S descriptions
  – Consistency checking
  – Automated publication with UDDI
  – Integrated Semantic discovery in UDDI
  – Automated generation of client code
**WS Development and invocation**

- **Web Service Development**
  - Implement Web service
  - Produce WSDL and OWL-S WS description
  - Deploy Web service
    - Advertise to available UDDI
    - Make service available for invocation

- **Web Service invocation on client side**
  - Find Web service in UDDI
  - Translate internal data representation to WS data representation
  - Invoke Web service consistently with specification of OWL-S Process Model

**All descriptions should fit together**
otherwise interaction with Web service fails
Overview OWL-S IDE

Developed tools and services:

- OWL-S Editor integrated with Eclipse
- Semantic UDDI
- OWL-S2UDDI Converter
- OWL-S API
- OWL-S VM

Key functionalities:

- Publish UDDI Port
- Generate OWL-S
- Generate WSDL
- OWL-S API provides easy processing in Java
- OWL-S VM provides an execution environment for OWL-S Web services
- Embed guided generation of WSDL and schematic OWL-S directly from Java exploiting Java2WSDL and WSDL2OWL-S tools

Automatic publication, inquiry and capability-based discovery with Semantic UDDI
OWL-S IDE Components

- **WSDL2OWL-S**
  map WSDL descriptions into OWL-S descriptions
- **OWL-S API**
  transform OWL-S code in an equivalent set of Java classes for easy processing
- **OWL-S Virtual Machine**
  control interaction with Web service consistently with Process Model and Grounding
- **OWL-S/UDDI translator**
  translate OWL-S Profiles in UDDI statements
- **Semantic UDDI**
  integrate UDDI Registry and OWL reasoning to facilitate discovery of Web services
Web Service Execution Environment (WSMX)

integrated Semantic Web service environment as the WSMO reference implementation

www.wsmx.org
WSMX Motivation

• Provide **middleware ‘glue’** for Semantic Web Services
  – Allow service providers focus on their business
• Provide a **reference implementation** for WSMO
  – Eat our own cake
• Provide an environment for **goal based** service discovery and invocation
  – Run-time binding of service requester and provider
• Provide a flexible **Service Oriented Architecture**
  – Add, update, remove components at run-time as needed
• Keep **open-source** to encourage participation
  – Developers are free to use in their own code
• Define **formal execution semantics**
  – Unambiguous model of system behaviour
WSMX Usage - P2P SWS Computing

complete the functionality for all the boxes
Design Principles

Strong Decoupling & Strong Mediation
autonomous components with mediators for interoperability

Interface vs. Implementation
distinguish interface (= description) from implementation (=program)

Peer to Peer
interaction between equal partners (in terms of control)

WSMO Design Principles == WSMX Design Principles
== SOA Design Principles
WSMX Architecture

WSMX Manager

WSM – Web Services Modelling Toolkit

WSM – Web Services Modelling Toolkit

WSMX Architecture

123
System Entry Points

**storeEntity**

- Service Provider
- Communication Manager (Requester Side)
- WSML
- Editor
- Parser
- Service Repository

Legend:
- WSMX components
- External entities
- Execution Flow
- Usage

**receiveGoal**

- Service Requester
- Communication Manager (Requester Side)
- WSML
- Adapter
- Parser
- Service Repository
- Matchmaker
- Selector
- Data Mediator

**receiveMessage**

- Service Requester
- Communication Manager (Requester Side)
- WSML
- Adapter
- Parser
- Service Repository
- Matchmaker
- Selector
- Data Mediator
- SOAP
Web Services Modelling Toolkit
Web Services Modelling Toolkit

- end-user / developer tool
- supports creation of WSMO element descriptions
  - domain ontologies
  - Web services descriptions
  - Goal specifications
  - Mediator descriptions
- communicates goals and service definitions to execution environments
WSMX @ Sourceforge.net

Project: Web Services Execution Environment: Summary

The Web Services Execution Environment (WSMX) is an execution environment for dynamic matchmaking, selection, mediation, invocation and interoperation of Semantic Web Services.

- Development Status: 3 - Alpha
- Intended Audience: Developers, Scientific/Research
- License: MIT License
- Programming Language: Java
- Topic: Distributed Computing

Project Owner: wms
Registered: 2004-06-29 12:46
Activity Percentile (last week): 37.64%
View project activity statistics
View list of RSS feeds available for this project

Latest File Releases

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
<th>Date</th>
<th>Notes / Monitor</th>
<th>Download</th>
</tr>
</thead>
<tbody>
<tr>
<td>toolkit</td>
<td>WSMX 0.1</td>
<td>March 16, 2005</td>
<td></td>
<td>Download</td>
</tr>
<tr>
<td>wsmx-components</td>
<td>WSMX Components 0.1.6</td>
<td>January 31, 2005</td>
<td></td>
<td>Download</td>
</tr>
<tr>
<td>wsmx-core</td>
<td>WSMX Core 0.01</td>
<td>July 30, 2004</td>
<td></td>
<td>Download</td>
</tr>
</tbody>
</table>

[View ALL Project Files]

Public Areas

WSMX v0.1 Released
monocen - 2005/03/16 12:17

Latest News

Semantic Web Services, HICSS 39, Kauai (Hawaii), 04 January 2006
WSMX Wrap Up

• WSMO as conceptual model (reference implementation)
• end to end functionality for executing SWS
• has a formal execution semantics
• real implementation
• open source code base at SourceForge
• event-driven component architecture
• growing functionality - developers welcome 😊
IRS-III:
A framework and platform for building Semantic Web Services

Stefania Galizia and Barry Norton
The Internet Reasoning Service is an infrastructure for publishing, locating, executing and composing Semantic Web Services.
Design Principles

- Ontological separation of User and Web Service Contexts
- Capability Based Invocation
- Ease of Use
- One Click Publishing
- Agnostic to Service Implementation Platform
- Connected to External Environment
- Open
- Complete Descriptions
- Inspectable
- Interoperable with SWS Frameworks and Platforms
Features of IRS-III (1/2)

- Based on Soap messaging standard
- Provides Java API for client applications
- Provides built-in brokering and service discovery support
- Provides *capability-centred* service invocation
Features of IRS-III (2/2)

• Publishing support for variety of platforms
  – Java, Lisp, Web Applications, Java Web Services

• Enables publication of ‘standard code’
  – Provides clever wrappers
  – One-click publishing of web services

• Integrated with standard Web Services world
  – Semantic web service to IRS
  – ‘Ordinary’ web service
IRS-III Architecture

LispWeb Server

WSMXML

Browser

Publishing Clients

Invocation Client

Publishing Platforms

SOAP

Browser Handler

Publisher Handler

Invocation Handler

WS Publisher Registry

OCML

WSMO Library

OWL(-S) Handler

OWL(-S)

SOAP Handler

Web Service

Java Code

Web Application

Publishing Platforms

LispWeb Server
Publishing Platform Architecture

Publishing Clients

SOAP

IRS-III Server

SOAP

Invocation Client

WS Service Registry

Service Registrar

Service Invoker

IRS-III Publishing Platform

HTTP Server

Web Service 1
Web Service 2
Web Service 3

Web Service 1
Web Service 2
Web Service 3
IRS-III/WSMO differences

- Underlying language OCML
- Goals have inputs and outputs
- IRS-III broker finds applicable web services via mediators
  - Used mediator within WS capability
  - Mediator source = goal
- Web services have inputs and outputs ‘inherited’ from goal descriptions
- Web service selected via assumption (in capability)
IRS-III Demo

(including Import from WSMX Toolset)

Stefania Galizia and Barry Norton
SWS Creation & Usage Steps

• Create a goal description
  – (e.g. exchange-rate-goal)
  – Add input and output roles
  – Include role type and soap binding

• Create a wg-mediator description
  – Source = goal
  – Possibly add a mediation service

• Create a web service description
  – Used-mediator of WS capability = wg-mediator above

• Specify Operation <-> Lisp function mapping in Choreography Grounding

• Publish against web service description

• Invoke web service by ‘achieve goal’
Multiple WS for goal

• Each WS has a mediator for used-mediator slot of capability
  – Some WS may share a mediator
• Define a kappa expression for assumption slot of WS capability
• Kappa expression format
  – (kappa (?goal) <ocml relations>)
• Getting the value of an input role
  – (wsmo-role-value ?goal <role-name>
Defining a Mediation Service

• Define a wg-mediator
• Source = goal
• Mediation-service = goal for mediation service
• Mediation goal
  – Mediation goal input roles are a subset of goal input roles
• Define mediator and WS as normal
Valid Relations

• Classes are unary relations
  – e.g. (country ?x)

• Slots are binary relations
  – e.g. (is-capital-of ?x ?y)

• Standard relations in base (OCML toplevel) ontology
  =, ==, <, >, member
European Currency Assumption

(kappa (?goal)
  (member
    (wsmo-role-value
      ?goal
      'has_source_currency)
    '(euro pound))))
Goal Based Invocation

Solve Goal
Goal -> WG Mediator -> WS/Capability/Used-mediator

Instantiate Goal Description
- Exchange-rate-goal
- Has-source-currency: us-dollars
- Has-target-currency: pound

Web Service Discovery
- European-exchange-rate-ws
- Non-european-exchange-rate-ws
- European-bank-exchange-rate-ws

WS -> Capability -> Assumption expression
- Web service selection
  - European-exchange-rate

Mediation
- Mediate input values
  - ‘$’ -> us-dollar

Invocation
- Invoke selected web service
  - European-exchange-rate
Hands-On Session
(with IRS III)

Barry Norton and Stefania Galizia
European Travel Scenario
European Travel Demo
IRS-III Hands On Task

- Develop an application for the European Travel scenario based on SWS. The application should support a person booking a train ticket between 2 European cities at a specific time and date.

- Create Goal, Web service and Mediator WSMO descriptions in IRS-III (european-travel-service-descriptions) for available services. Your descriptions should choose a specific service depending on the start and end locations and the type of traveller. Use the assumption slot to do this.

- Publish available lisp functions against your descriptions.

- Invoke the web services.

- Solution to be shown at the end of this session.
Tutorial Setup

 IRS Server (3000)
  - Domain Models
  - Web Service WSMO Descriptions
    + Registry of Implementors
  - Goal WSMO Descriptions
    + SOAP Binding
  - Mediator WSMO Descriptions

 IRS Lisp Publisher

 WSMX

 Travel Services (3001)

 IRS-III Knowledge Model Browser & Editor
Travel Related Knowledge Models
Is-in-country <city> <country> e.g.
(is-in-country berlin germany) -> true

(student <person>) -> true, for john matt michal
(business-person <person>) -> true, for liliana michael
Goals

1- Get train timetable
   – Inputs: origin and destination cities (city), date (date-and-time, e.g. (18 4 2004))
   – Output: timetable (string)

2- Book train
   – Inputs: passenger name (person), origin and destination cities, departure time-date (list-date-and-time, e.g. (20 33 16 15 9 2004))
   – Output: booking information (string)
Services

• 1 service available for goal 1
  – No constraints

• 6 services available for goal 2
  – As a provider write the constraints applicable to the services to satisfy the goal (assumption logical expressions)

• 1 wg-mediator mediation-service
  – Used to convert time in list format to time in universal format
Service constraints

• Services 2-5
  – Services for (origin and destination) cities in determined countries

• Service 4-5
  – Need a mediation service to map goal time-date to service time-date

• Services 6-7
  – Services for students or business people in Europe
Available Functions (1/3)

1- get-train-times

*paris* *london* *(18 4 2004)*
"Timetable of trains from PARIS to LONDON on 18, 4, 2004
5:18
…23:36"

2- book-english-train-journey

*christoph milton-keynes* *london* *(20 33 16 15 9 2004)*
"British Rail: CHRISTOPH is booked on the 66 going from MILTON-KEYNES to
LONDON at 16:49, 15, SEPTEMBER 2004. The price is 169 Euros."

3- book-french-train-journey

*sinuhe* *paris* *lyon* *(3 4 6 18 8 2004)*
"SNCF: SINUHE is booked on the 511 going from PARIS to LYON at 6:12, 18,
AUGUST 2004. The price is 27 Euros."
4- book-german-train-journey
christoph berlin frankfurt 3304251200
"First Class Booking German Rail (Die Bahn): CHRISTOPH is booked on the 323 going from BERLIN to FRANKFURT at 17:11, 15, SEPTEMBER 2004. The price is 35 Euros."

5- book-austrian-train-journey
sinuhe vienna innsbruck 3304251200
"Austrian Rail (OBB): SINUHE is booked on the 367 going from VIENNA to INNSBRUCK at 16:47, 15, SEPTEMBER 2004. The price is 36 Euros."
6- *book-student-european-train-journey*

`john london nice (3 4 6 18 8 2004)`

"European Student Rail Travel: JOHN is booked on the 916 going from LONDON to NICE at 6:44, 18, AUGUST 2004. The price is 94 Euros. "

7- *book-business-european-train-journey*

`liliana paris innsbruck (3 4 6 18 8 2004)`

"Business Europe: LILIANA is booked on the 461 going from PARIS to INNSBRUCK at 6:12, 18, AUGUST 2004. The price is 325 Euros."

8- **mediate-time (lisp function) or JavaMediateTime/mediate (java)**

`(9 30 17 20 9 2004)`

3304686609
Example: Goal

In IRS-III or In WSMT
Example: Mediator

In IRS-III

or

In WSMT
Example: Service

In IRS-III

or

In WSMT
Example: Publishing
Tips

• Order matters for input roles
  – Input roles in goal must match order of arguments to function
• Need to specify both input roles and output role
• Be careful with soap binding
  – sexpr as default
  – String for one line output
  – Use xml for multiple line output
• Input roles for web services inherited from goal
• Slot names can not be the same as class names
• Goal <-> web service linking mediator in the capability used mediators
Closing, Outlook, References, Acknowledgements
Tutorial Wrap-up

• The **targets** of the presented tutorial were to:
  – understand aims & challenges within Semantic Web Services
  – understand OWL-S and WSMO:
    • design principles & paradigms
    • ontology elements
• .. an **overview of ‘hot topics’** within the Semantic Web and Semantic Web Services
• .. OWL-S and WSMO **Tools and System Presentation**
• .. do-it-yourself **Hands-On Session**

=> you should now be able to correctly **assess emerging technologies & products** for Semantic Web Services and utilize these for your future work
OWL-S and WSMO

• North-American and European initiatives with converging aims
• Offer a SWS platforms to be used by B2C and B2B applications
• Provide a backbone for advanced integration and automation of industrial and business processes
• Are the most developed SWS technologies up to now available to be used in commercial and industrial applications
• Developments towards refining and interconnecting them
Future work – OWL-S

- OWL-S is close to conclusion, but a few issues still need to be addressed
  - An exception mechanism is still missing
  - There is a need of an exec instruction for loading and executing Process Models dynamically
  - A new Grounding for WSDL 2 should be developed

- Additional issues that OWL-S does not address
  - Security and Policies are not directly expressed in OWL-S yet
  - There are no facilities for Contracting and agreement
  - There are no facilities for Web service management
Future work – OWL-S (2)

• Standardization
  – The OWL-S coalition is planning to submit a W3C note to draw attention and create momentum for W3C standardization activities on Semantic Web services
  – Members of the OWL-S coalition are already active in standardization committee such as UDDI, WSDL 2 and WS Coordination

• The Future of OWL-S
  – OWL-S is nearing its completion and it will converge in the results of the SWSI working group or future standardization activities
  – The OWL-S coalition plans to remain in existence to maintain and further develop the language if needed
Future work - WSMO

• Further develop and consolidate concepts and implementation aspects of WSMO, WSML and WSMX
  – Choreography and orchestration
  – Business process execution
  – Web services composition
  – Process and protocol mediation

• Open to new ideas, contributions and suggestions
Future Work WSMO (2)

- Standardization …
- WSMO & WSMX – applied in several case studies within EU funded projects
- WSMO Studio development
- WSMX v2 to be release in November
Future Work IRS

• IRS III further integration with WSMX toolset on-going
• IRS-III to be applied in:
  – Business Processes Modelling (w/ SAP in DIP, and new EU project SUPER)
  – Geographical Information Systems (DIP project)
  – Biomed Modelling (new EU project Living Human Digital Library)
  – eLearning (new EU project LUISA)
Future Work IRS (2)

- IRS orchestration and choreography to be extended to three-level model:
  - Graphical language: **UML Activity Diagrams**
  - Workflow language: **Cashew**
    - extends OWL-S
    - aligns with Workflow Patterns
    - expresses choreography, as well as orchestration
  - Executable language: **Ontologized Abstract State Machines**
Beyond OWL-S and WSMO

- Although OWL-S and WSMO are the main initiatives on Semantic Web services, they are not the only activities
- Semantic Web Services Interest Group
  - Interest group founded at W3C to discuss issues related to Semantic Web Services (http://www.w3.org/2002/ws/swsig/)
- SWSI: International initiative to push toward a standardization of SWS (http://www.swsi.org)
- WSDL-S: Semantic Annotation of WSDL interfaces
- Semantic Web services are entering standardization
  - W3C working groups currently starting
  - OASIS working groups currently starting
=> eventually major influence on next generation Web technology
References OWL-S

• The main repository of papers on OWL-S is at http://www.daml.org/services/owl-s/pub-archive.html that contains many papers produced by the coalition as well as from the community at large

• The main source of information on OWL-S is the Web site http://www.daml.org/services/owl-s

• The rest of this section will report what we believe to be the most influential papers on OWL-S as well as paper referred in this tutorial
References OWL-S

- **Fundamental**
  


References OWL-S

• Discovery


References OWL-S

• Composition and Invocation


References OWL-S

• Formal Models and Verification

Anupriya Ankolekar, Massimo Paolucci, and Katia Sycara


References OWL-S

• Policies and Security

Ronald Ashri, Grit Denker, Darren Marvin, Mike Surr ridge, Terry Payne, 
Semantic Web Service Interaction Protocols: An Ontological 
Approach, 3rd International Semantic Web Conference (ISWC2004), 
Hiroshima, Japan

Lalana Kagal, Grit Denker, Tim Finin, Massimo Paolucci, Naveen 
Srinivasan and Katia Sycara, "An Approach to Confidentiality and 
Integrity for OWL-S", forthcoming in Proceedings of AAAI 2004 Spring 
Symposium.

Grit Denker, Lalana Kagal, Tim Finin, Massimo Paolucci, Naveen 
Srinivasan and Katia Sycara, "Security For DAML Web Services: 
Annotation and Matchmaking" In Proceedings of the Second 
International Semantic Web Conference (ISWC 2003), Sandial Island, 
FL, USA, October 2003, pp 335-350.
References OWL-S

• Applications


Aabhas V Paliwal, Nabil Adam, Christof Bornhövd, and Joachim Schaper Semantic Discovery and Composition of Web Services for RFID Applications in Border Control In Proceedings of Workshop on Semantic Web Services: Preparing to Meet the World of Business Applications (ISWC 2004)


References WSMO

• The central location where WSMO work and papers can be found is WSMO Working Group: http://www.wsmo.org

• WSMO languages – WSML Working Group: http://www.wsml.org

• WSMO implementation
  – WSMX working group: http://www.wsmx.org
  – WSMX open source can be found at: https://sourceforge.net/projects/wsmx/
References WSMO


References WSMO

References WSMO

• [Stencil Group] - www.stencilgroup.com/ideas_scope_200106wsdefined.html
References Discovery


References Discovery


References IRS III


These papers and software downloads can be found at: http://kmi.open.ac.uk/projects/irs
Acknowledgements

We would like to acknowledge the contribution of the past and present members of the OWL-S coalition for their hard work in the development of the language. Furthermore, we would like to thank the community at large for contributing to tools and ideas.

Furthermore, we would like to thank to all the members of the WSMO, WSML, and WSMX working groups for their advice and input into this tutorial.

Special thanks to Sheila McIlraith, Craig Schlenoff, Daniel Elenius and Naveen Srinivasan for providing slides and suggestions on this tutorial.

Slide design by Roberta Hart-Hilber, DERI Austria
Acknowledgements

The development of **OWL-S** has been funded almost exclusively by the **DAML DARPA** program.

The **WSMO working groups** are funded by the European Commission under the projects **DIP**, **Knowledge Web**, **SEKT**, **SWWS**, and **ASG**; by **Science Foundation Ireland** under the **DERI-Lion** project; and by the Vienna city government under the **FIT-IT Programme** in the projects **RW²** and **TCP**.

IRS development is funded by the European Commission under the **DIP** project, and formerly **IBROW**, and by the UK EPSRC under the **AKT** project, and formerly **MIAKT**.