SESA Tutorial
Semantically Enabled Service-Oriented Architectures

DERI Innsbruck
Tutorial Team

Michael Stollberg

Darko Anicic

Omair Shafiq
Agenda

• **Day 1: Introduction**
  – Semantic Web, Ontologies, Semantic Web Services
  – The WSMO Framework

• **Day 2: Logics & Reasoning**
  – Ontologies & Reasoning
  – The WSML Language
  – Hands-on Session

• **Day 3: SESA Environments**
  – Semantic Web Service Execution Environments
  – The WSMX System
  – Hands-on Session

• **Day 4: Applications & SUPER project**
  – WSMO L X in real world applications
  – The SUPER Project
Mission

• The mission of the Digital Enterprise Research Institute (DERI) is to establish semantics as a core pillar of modern computer engineering

Semantically Enabled Service-oriented Architectures - SESA
Semantic Web Services - SWS
Semantic Web - SW
Main Research Fields

• Semantic Web & Ontologies
  – ontology languages
  – reasoning infrastructures
  – semantic data integration

• Semantic Web Services
  – semantic WS annotation
  – SWS technologies: automated discovery, composition, mediation, selection, compatibility analysis
  – automated WS execution
  – Web Services & SOA

=> Semantically Enabled Service-Oriented Architectures
The Web Service Modeling Ontology

Objectives that a client wants to achieve by using Web Services

Formally specified terminology used by all other components

Ontologies

Goals

Web Services

Mediators

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

Connectors between components with mediation facilities for handling heterogeneities
Day I

Introduction

Semantic Web

Semantic Web Services

WSMO

Michael Stollberg
Agenda

• **The Vision**
  – The Semantic Web
  – Web Services
  – Semantic Web Services

• **The Web Service Modeling Ontology WSMO**
  – a comprehensive framework for SESA
  – elements & languages
  – central techniques

LUNCH BREAK

• **Other Approaches**
  OWL-S, SWSF, WSLD-S, SAWSDL

• **Outlook for Next Days**
  – Logics & Reasoning
  – SESA Development Environments
Next Generation of the WWW

Dynamic

Web Services
UDDI, WSDL, SOAP

Semantic Web Services

Static

WWW
URI, HTML, HTTP

Semantic Web
RDF, RDF(S), OWL

making semantics real.
The Semantic Web

- next generation of the Internet (augmentation of the WWW)
- information has machine-processable and machine-understandable semantics
- ontologies as base technology for semantic interoperability

Semantic bridges:

- RDF
- OWL
- Web Appl.
- XML
- DB
Ontology Definition

Formal explicit specification of a shared conceptualization

- Unambiguous terminology definitions
- Machine-readability with computational semantics
- Conceptual model of a domain (ontological theory)
- 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

Instance
individual in the domain

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

```
Ann memberOf student
  name = Ann Lee
  studentID = 12345
```
Ontology Languages

Requirements
- expressivity
- reasoning support
- web compliance

W3C Semantic Web Language Layer Cake
revised version, Tim-Berners-Lee 2005
Ontology Technology

• Ontology Reasoning
  + advanced information processing
    – special requirements
      • large scale knowledge handling
      • fault-tolerant
      • stable & scalable inference machines

• Ontology Management
  – (collaborative) editing and browsing
  – storage and retrieval
  – versioning and evolution support
Ontology-Based Data Integration

- Ontology Integration Techniques

- data integration on semantic level (domain independent)
- semi-automatic
  - human intervention needed for “integration decision
  - graphical support for ontology mapping as central technique
Web Services & SOA

• Web Service = program accessible over the Web

• Service-Oriented Architecture (SOA):
  – use Web services as basic building blocks
  – dynamically find & invoke those Web services that allow to solve a particular request

• Web Service Technologies:
  1. **WSDL**  Web Service Description Language
  2. **SOAP**  XML data exchange protocol for the Web
  3. **UDDI**  registry for Web Services
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)
• Simple Object Access Protocol
• W3C Recommendation

XML data transport:
- sender / receiver
- protocol binding
- communication aspects
- content
- Universal Description, Discovery, and Integration Protocol
- OASIS driven standardization effort

Registry for Web Services:
- provider
- service information
- technical access
The Web Service Usage Process

- **Repository**
  - find usable Web Service

- **WSDL**
  - describes

- **Consumer**
  - points to

- **Web Service**
  - WS usage via message exchange
  - SOAP

WSDL describes a Web Service, which the Consumer can access via SOAP messages. The Consumer searches the Repository to find a usable Web Service.
Deficiencies 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

=> initial Web Service Technology Stack failed to realize the SOA Vision
Semantic Web Services

• automate Web Service technologies by
  1. rich, formal annotation of Web Services
  2. inference-based techniques for automated discovery, composition, mediation, execution of Web Services

• integration with the Semantic Web
  – ontologies as data model
  – Web Services as integral part

• Semantically Enabled SOA (SESA):
  – also semantically describe client requests
  – mediation techniques as integral part
  ⇒ semantically enhance & automate the complete SOA life cycle
Semantic Web Services

a) Web Service Description Structure

- Interface
- Web Service Implementation (not of interest in Web Service Description)
- XML

b) Semantic Web Service Description Structure

- Non-functional
- Functionality
- Web Service Implementation (not of interest in Web Service Description)
- Interface
- Aggregation
- Ontology
Goal-driven Web Service Usage

Client

objective / problem to be solved

client-system interaction

Goals

formal objective description

discovery, composition, mediation

Semantics / SWS

Ontology

SWS description

Mediator

Web Services & Resources

Web Service

Internet

making semantics real.
SESA = Automated Goal Solving with WS

**GOAL**

- **Discoverer**
  - if: usable
  - if: composition possible
    - uses
      - **Composer**
  - else: not solvable
    - matchmaking R with all WS

- **Data Mediator**
  - uses
  - **Process Mediator**
    - if: compatible
    - uses
      - **Selection & Ranking**
    - else: try other WS

- **Behavioral Conformance**
  - if: composition possible
    - uses
      - **Executor**
  - if: executable
    - information lookup for particular service

- **Service Repository**
  - if: successful
  - submission
The WSMO Framework
Web Service Modeling Ontology (WSMO)

- Comprehensive Framework for SESA
  - top level notions = SESA core elements
  - conceptual model + axiomatization
  - ontology & rule language

- International Consortium (mostly European)
  - started in 2004
  - 78 members from 20 organizations
  - W3C member submission in April 2005

www.wsmo.org
WSMO Top Level Notions

Objectives that a client wants to achieve by using Web Services

Formally specified terminology used by all other components

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

Connectors between components with mediation facilities for handling heterogeneities

**W3C submission 13 April 2005**
WSMO Working Groups

- Conceptual Model & Axiomatization for SWS
- Formal Language for WSMO
- Ontology & Rule Language for the Semantic Web
- Execution Environment for WSMO

www.wsmo.org
The Web Service Modeling Ontology

Objectives that a client wants to achieve by using Web Services

Formally specified terminology used by all other components

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

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)
Specification Language: WSML

WSML-DL \[\rightarrow\] First-Order Logic (with nonmonotonic extensions) \[\rightarrow\] WSML-Full

WSML-Core \[\leftarrow\] Description Logics \[\rightarrow\] WSML-Flight \[\rightarrow\] WSML-Rule

Logic Programming
The Web Service Modeling Ontology

Objectives that a client wants to achieve by using Web Services

Formally specified terminology used by all other components

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

Connectors between components with mediation facilities for handling heterogeneities
WSMO Web Service Description

**Capability**
- Advertising of Web Service
- Support for WS Discovery

**Non-functional Properties**
- complete item description
- quality aspects
- Web Service Management

- Advertising of Web Service
- Support for WS Discovery

**DC + QoS + Version + financial**

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

**Choreography --- Service Interfaces --- Orchestration**

- client-service interaction interface for consuming WS
- external visible behavior
- communication structure
- ‘grounding’

- realization of functionality by aggregation
- functional decomposition
- WS composition
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
- Shared Variables: scope is entire capability
- 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)
Example VTA Web Service

- Web service for booking tickets or complete trips
- capability precondition (WSML)

```plaintext
capability  VTACapability
sharedVariables  {?item, ?passenger, ?creditCard, ?initialBalance, ?price}
precondition
definedBy
  exists  ?reservationRequest
    (?reservationRequest[
      reservationItem  hasValue  ?item,
      passenger  hasValue  ?passenger,
      payment  hasValue  ?creditcard]
    memberOf  tr#reservationRequest  and
    (?item  memberOf  tr#trip  or  ?item  memberOf  tr#ticket)  and
    ?passenger  memberOf  pr#person  and
    ?creditCard  memberOf  po#creditCard  and
    (?creditCard[type  hasValue  po#visa]  or
    ?creditCard[type  hasValue  po#mastercard])  )  .
```
Example VTA Web Service

- WSMO capability assumption:
  - the provided credit card is valid
  - the balance of the credit card before executing the service is higher than the price of the reservation (= purchased item) that is retrieved after executing the Web service.

assumption
definedBy
  po#validCreditCard(?creditCard) and
  ?creditCard[balance hasValue ?initialBalance] and
  (?initialBalance >= ?price) .
Example VTA Web Service

- capability description (post-state)

```log
postcondition
  definedBy
    exists ?reservation(?reservation[
      reservationItem hasValue ?item,
      price hasValue ?reservationPrice,
      customer hasValue ?passenger,
      payment hasValue ?creditcard]
     memberOf tr#reservation and
      ?price memberOf tr#price).

effect
  definedBy
    ?creditCard[po#balance hasValue ?finalBalance] and
    (?finalBalance = (?initialBalance - ?price)).
```
Choreography & Orchestration

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

**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
interface for interaction with aggregated Web Services

- decomposition of service functionality
- other Web services consumed via their choreography interfaces
Ontologized Abstract State Machines

• Description
  Vocabulary:
  • ontology constructs used in service interface description
  • usage for information interchange: in, out, shared, controlled

States:
  • a stable status in the information space
  • defined by attribute values of ontology instances

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

• Usage:
  – partners A, B commence interaction with empty Ω_A, Ω_B
  – Ω_A, Ω_B are updated via Guarded Transitions in each state
  – interaction termination state when A, B have no further transition rules
Example Hotel Web Service

- choreography interface (state signature)

```xml
interface htl#BookHotelInterface
coreography
stateSignature
importsOntology htl#simpleHotelOntology
in
  htl#HotelRequest withGrounding "http://...",
  htl#HotelConfirm withGrounding "http://...",
  htl#HotelCancel withGrounding "http://...
out
  htl#HotelNotAvailable withGrounding "http://...",
  htl#HotelOffer withGrounding "http://....
shared
  htl#Hotel,
  htl#HotelAvailable,
  htl#HotelBooked
```
Example Hotel Web Service

- choreography interface (transition rules)

```plaintext
ctl_state {htl#start, htl#offerMade, htl#noAvail, htl#confirmed, htl#cancelled}
transitionRules
  if (ctl_state = htl#start) then
    forall {?req, ?date, ?loc, ?client} with
      ?req[trv#date hasValue ?date, trv#location hasValue ?loc,
      htl#client hasValue ?client] memberOf htl#HotelRequest
      do
        add(htl#offer(?req)[trv#date hasValue ?date,
        trv#hotelName hasValue ?name, trv#location hasValue ?loc,
        htl#client hasValue ?client] memberOf htl#HotelOffer)
        ctl_state := htl#offerMade
      |
        add(htl#notAvailable(?req)[trv#date hasValue ?date,
        trv#location hasValue ?loc] memberOf htl#HotelNotAvailable)
        ctl_state := htl#noAvail
    endForall
  endIf
```
The Web Service Modeling Ontology

Objectives that a client wants to achieve by using Web Services

Formally specified terminology used by all other components

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

Connectors between components with mediation facilities for handling heterogeneities
WSMO Goals

• formal description of client objective
  – problem that a client wants to achieve by using WS
  – abstracting from technical details

• purpose
  1. facilitate problem-oriented WS usage
  2. dynamic WS usage
  \[\Rightarrow \text{central client side element for SESA}\]

• WSMO 1.2 Goal description:
  – requested capability (WHAT)
  – requested choreography (restrictions on comm. behavior)
  – requested orchestration (= goal decomposition)
Goal Model (Proposed Revision)

Composite Goal

- subGoals: goal
- workflow: orchestration

Goal

- domain: ontology
- objective: capability

Goal Instance

- template: goal
- input: inputbinding

WG Mediator

- source: goal
- target: webService
- usability: matching degree
- clientInterface: choreography

desired workflow

instantiation

automated Web service usage
WSMO Goals

Client-Side

Goal Template
generic objective description

Goal Instance
concrete input

Client

defines

Service-Side

functional

(Web) Service Implementation
(not of interest here)

behavioral

service detection

service usage

Domain Knowledge

Ontology

Ontology

Ontology

Ontology
### Basic Goal

- **Goal**: buy train ticket in Germany
  - origin: o, destination: d
  - date-time: dt

- **Client Interface**: goal instance with inputs:
  - o = Munich, d = Berlin
  - dt = 20070319-1030

- **WG Mediator**

- **DB Ticketing**

**Design Time**
- instantiates
- defines

**Runtime**
- executes
- client interface
Composite Goal

Flight-hotel booking with desired workflow

Goal

Flight Request

Hotel Request

Book Flight

Book Hotel

if hotel = Ø
flight.outwardArrival = hotel.arrival

if flight = Ø

flight information

hotel information

Interface (Chor.)
1) get request
2) provide offer
3) receive selection
4) send confirmation

Flight WS

Capability

Orch.

Hotel WS

Capability

Orch.
The Web Service Modeling Ontology

Objectives that a client wants to achieve by using Web Services

- Formally specified terminology used by all other components
- Semantic description of Web Services:
  - Capability (functional)
  - Interfaces (usage)

Connectors between components with mediation facilities for handling heterogeneities
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 resolve mismatches independent of content
- mediation cannot be fully automated (integration decision)

Levels of Mediation in WSMO:
- Representation: heterogeneous Languages & Protocols
- Data Level: heterogeneous Data Sources
- Functional Level: heterogeneous Functionalities
- Process Level: heterogeneous Communication Processes
Mediation Definition (1) has Mediation Services
Mediation Service
- indirectly (via a Goal)
- directly
Target Component

Source Component 1..n
Source Component

Specification layer
Execution layer

Mediation Services

Mediator Structures
- Specification layer
- Execution layer
WSMO Mediators Overview

- **OO Mediator**
  - Input: O
  - Output: O / G / WS / M

- **GG Mediator**
  - Input: G
  - Output: G

- **WG Mediator**
  - Input: WS xor G
  - Output: WS xor G

- **WW Mediator**
  - Input: WS
  - Output: WS

**Legend**
- Technique used
- Imports / reuses
- Correlation

**Data Level Mediation**
- Terminology
- Representation & Protocol

**Process Level**
- Communication

**Δ-Relation Mediation**
- Process Level (Communication)
- Process Level (Cooperation)
Central SESA Techniques
SESA Tutorial
Seoul, 10 - 14 September
2007

SESA Process

GOAL

- if: successful
  - submission
- else: not solvable
  - matchmaking R with all WS

Discoverer

- if: usable
- if: composition possible

Data Mediator

Process Mediator

Behavioral Conformance

Selection & Ranking

Composer

Service Repository

Executor

- if: execution error
- else: try other WS

uses

uses

uses

uses

composition (executable)

information lookup for particular service
Techniques

- **Discovery**
  find candidate WS to solve a Goal

- **Selection & Ranking**
  select best candidate / determine a priority list

- **Composition**
  combine several WS to solve a Goal

- **Behavioral Compatibility**
  ensure that interaction can take place

- **Mediation**
  resolve & handle possibly occurring heterogeneities

- **Execution**
  automatically invoke & consume WS to solve a Goal
Web Service Discovery

detect directly usable Web services out of available ones

Techniques:

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
Semantic Matchmaking

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:
\[ G, WS, O, M \models \forall x. (G(x) \leq WS(x)) \]

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

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

Selection & Ranking

**Selection**

determine best candidate out of discovered WS

**Ranking**
determine a priority list of discovered WS

- after “functional” discovery
- Criteria:
  - Quality of Service (security, robustness, availability)
  - context (regional, business / social communities)
  - preferences and policies
  - financial
  - …
combine several Web services for solving a request

- applied if no directly usable Web service exists to solve a goal

- Types of Composition Techniques:
  1. functional: sequence of WS
  2. behavioral: interaction in composition is possible

- Integrated Web Service Composition:
  1. skeleton by functional composition
  2. refinement + executable code by behavioral composition

Procedure:

- directly usable WS (discovery)?
  - yes
    - composition (functional)
      - a)
      - b)
      - no
        - abort
  - no
    - composition (functional)
      - skeleton
      - composition (behavioral)
      - no
        - executable composition
        - abort
      - yes
        - executable composition
        - abort
Behavioral Compatibility

determine whether partners can interact successfully

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

\[ \Omega_G(\omega_0) = \{\emptyset\} \]
if \( \emptyset \) then request
\[ \Omega_G(\omega_1) = \{\text{request(out)}\} \]
if cnd1(offer) then changeReq
\[ \Omega_G(\omega_{2a}) = \{\text{offer(in), changeReq(out)}\} \]
if cnd2(offer) then order
\[ \Omega_G(\omega_{2b}) = \{\text{offer(in), order(out)}\} \]
if conf then \( \emptyset \)
\[ \Omega_G(\omega_3) = \{\text{offer(in), conf(in)}\} \]

VTA Behavior Interface

Start
\[ \Omega_{VTA}(\omega_0) = \{\emptyset\} \]
if request then offer
\[ \Omega_{VTA}(\omega_1) = \{\text{request(in), offer(out)}\} \]
if changeReq then offer
\[ \Omega_{VTA}(\omega_{2a}) = \{\text{changeReq(in), offer(out)}\} \]
if order then conf
\[ \Omega_{VTA}(\omega_{2b}) = \{\text{order(in), conf(out)}\} \]

Termination

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)

**Flight WS Behavior Interface**

Start

(VTA, FWS)

if request then offer

if order then confirmation

Termination

(VTA, FWS)

**Hotel WS Behavior Interface**

Start

(VTA, HWS)

if request then offer

if order then confirmation

Termination

(VTA, HWS)
Mediation

• Heterogeneity as inherent characteristic of the Web:
  – heterogeneous terminology
  – heterogeneous languages / formalisms
  – heterogeneous functionalities
  – heterogeneous communication protocols and business processes

• WSMO identifies Mediators as top level element
  – levels of mediation: data, functional, protocol, processes
  – WSMO Mediator types

• Approach: declarative, generic mismatch resolution
  – classification of possible & resolvable mismatches
  – mediation definition language & mediation patterns
  – execution environment for mediation definitions
Data Mediation Techniques

• Ontology Integration Techniques

- Ontology Mapping
- Ontology Alignment
- Ontology Merging

- semi-automatic
  - human intervention needed for “integration decision
  - graphical support for ontology mapping as central technique
Ontology O1

Human
  - name

Adult
Child

Ontology O2

Person
  - name
  - age

michael memberOf Person
  - name = Michael Stollberg
  - age = 28

classMapping(unidirectional o2:Person o1.Adult
  attributeValueCondition(o2.Person.age >= 18))

this allows to transform the instance ‘michael’ of concept person in
ontology O2 into a valid instance of concept ‘adult’ in ontology O1
Process Level Mediation

- not a priori compatible behavior interfaces for communication & information interchange

- partially resolvable by “process mediation patterns”
Patterns for Resolvable Mismatches
can resolve about 80% of process level mismatches
Automated Web Service Execution

Client Interface (semantic) → Interaction (semantic) → Web Service Interface (semantic) → Execution (via grounding) → Web Service Interface (syntactic) → WWW (SOAP) → WSDL

Communication Space

WWW (SOAP)

Web Service
Other SWS Approaches
Upper Ontology for Web Service Descriptions

- capability description (IOPE)
- non-functional properties
- usage: (1) WS advertisement, (2) WS request formulation

- specification of service access information
- builds upon WSDL to define message structure and physical binding layer
- specifies communication protocols & language, transport mechanisms, etc.

- describes internal processes of the service
- defines service interaction protocol for (a) consumption and (b) WS interaction
- process types: simple, atomic, composite
- specifies: (1) abstract messages (ontological content), (2) control flow constructs, (3) perform construct
• **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:

**OWL-S Profile** ≈ **WSMO capability + non-functional properties**

**OWL-S Process Model** ≈ **WSMO Service Interfaces**

**OWL-S Grounding** ≈ **current WSMO Grounding**

• Goals and Mediators not in scope
• deficiencies in Service Model (process description model / language not adequate) => SWSF
WSML aims at overcoming deficiencies of OWL
• Process Model for Web Services (FLOWS)
• although self-contained, commonly understood as extension of OWL-S / refinement of Service Model

<table>
<thead>
<tr>
<th>Module</th>
<th>Explanation</th>
<th>Major Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOWS-Core</td>
<td>basic notions of services as activities composed of atomic activities</td>
<td>Service AtomicProcess composedOf message channel</td>
</tr>
<tr>
<td>Control Constraints</td>
<td>common workflow-style process constructs, including OWL-S process model concepts.</td>
<td>Split Sequence Unordered Choice Iterate IfThenElse RepeatUntil</td>
</tr>
<tr>
<td>Ordering Constraints</td>
<td>allow specification of activities defined by sequencing properties of atomic processes</td>
<td>OrderedActivity</td>
</tr>
<tr>
<td>Occurrence Constraints</td>
<td>support for nondeterministic activities within services</td>
<td>OccActivity</td>
</tr>
<tr>
<td>State Constraints</td>
<td>specify activities that are triggered by states (of an overall system)</td>
<td>TriggeredActivity</td>
</tr>
<tr>
<td>Exception Constraints</td>
<td>basic infrastructure for modeling exceptions</td>
<td>Exception</td>
</tr>
</tbody>
</table>
Semantic annotation of WSDL descriptions
1. annotate XML Schema with domain ontology

```xml
<xsl:element name="processPOResponse" type="xs:string
  wssem:modelReference="POOntology#OrderConfirmation"/>
```

2. pre-conditions & effects for operations

```xml
<interface name="PurchaseOrder">
  <operation name="processPurchaseOrder" pattern=wsdl:in-out>
    <input messageLabel="processPORequest" element="tns:processPORequest"/>
    <output messageLabel="processPOResponse" element="processPOResponse"/>
    <wssem:precondition name="AccExistsPrecond" wssem:modelReference="onto#AccountExists">
      <wssem:effect name="ItemReservedEffect" wssem:modelReference="onto#ItemReserved"/>
    </wssem:precondition>
  </operation>
</interface>
```

3. WS categorization by ontology-based keywords

```xml
<wssem:category name="Electronics"
  taxonomyURI="http://www.naics.com/" taxonomyCode="443112" />
```
• „Semantic Annotation for WSDL“

• Similar to WSDL-S:
  – semantic annotation of XML-Schema
  – semantic annotation of WSDL (interface & operations)

• … but more limited:
  – agnostic to ontology languages
  – references only to concepts

W3C Recommendation 28 August 2007
Commonalities & Differences

• similar ontological structure for WS descriptions
  – Functional Descriptions (preconditions & effects)
  – Behavioral Descriptions (consumption and interaction)
  – Grounding to WSDL (automated execution)

• central conceptual differences
  – formal models for capabilities
  – interfaces vs. business process
  – behavioral aspects:
    state-based ⇔ process models ⇔ operation-level capabilities

• WSMO defines “core elements for SESA” while all others are only concerned with describing Web Services
Outlook Day 2:
WSML Language & Reasoning

Michael Stollberg
The WSML Language

- specification language for WSMO elements
- ontology language with 5 variants

W3C Semantic Web Language Layer Cake
revised version, Tim-Berners-Lee 2005

WSML Language Family
WSMO4J – an API for WSMO

WSMO4J (API)
1. WSMO data model (OGWM)
2. Getter & Setter Methods
3. WSML Parser & Serializer

Discovery
Composition
Mediation
Executor

WSML2Reasoner

Reasoner
...

Resource Manager

Discovery
Composition
Mediation
Executor
Outlook Day 3: The WSMX System (WSMO Reference Implementation)
WSMX - The Web Service Execution Environment

- A Service Oriented Architecture
- Reference implementation of SESA and WSMO
- open source (LGPL): http://sourceforge.net/projects/wsmx/

[Diagram of the WSMX architecture with layers and components such as Ontologies, Applications, Developer Tools, Discovery, Adaptation, Composition, Choreography, Mediation, Grounding, Fault Handling, Monitoring, Formal Languages, Reasoning, Storage and Communication.]
Event-based Implementation
WSMX Functionalities

- **“Achieve a goal”**
  - parse goal
  - WS detection (all necessary functional components)
  - invoke WS

- **“Get Web Services”**
  - find WS with respect to certain properties
  - for WS repository management

- **“Get a Web Service”**
  - ranking / selection for "get the best"
  - non-functional aspects

- **“Invoke a Web Service”**
  - grounding to WSDL
  - WS execution via SOAP
WSMT – Developer Toolkit
Other WSMO Tools

- WSMO Studio (DERI & OntoText)
  - form-based WSMO editing
  - integrated reasoners & repository
  - extensions, e.g. BPMO

- WSML Validator
  - syntax validator for WSML
  - Web Interface
  - provided as Web service

- Ontology Technology
  - WSML Ontology Management Suite (DOME, OMS)
  - Data Mediator (stand-alone)

_all open source (LGPL licence)_
References
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 Foundations


References Semantic Web Services


References SWS: W3C Submissions

OWL-S

WSMO [see also www.wsmo.org]

SWSF

WSDL-S
References Discovery


References Discovery


References Composition


References Mediation


Acknowledgements

The WSMO working groups are funded by the European Commission under the projects DIP, Knowledge Web, SEKT, SWWS, ASG, and SUPER; 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 TSC.

We dedicate thanks to all the members of the WSMO, WSML, and WSMX working groups for their advice and input into this tutorial.