Semantic Web Services

Tutorial

AAAI 2006 Tutorial Forum

Michael Stollberg
Emilia Cimpian

Liliana Cabral
John Domingue
Contents

Morning Session
• Part I: Introduction to Semantic Web Services
• Part II: Semantic Web Service Frameworks
• Part III: Semantic Techniques for Automated Web Service Usage
• Part IV: Standardization, Market Prospects, Future Issues

Afternoon Session
• Part V: Web Service Execution Environments (WSMX and IRS)
• Part VI: Hands-On Session

PART I:
Introduction to Semantic Web Services

Michael Stollberg
"Semantic differences remain the primary roadblock to smooth application integration, one which Web Services alone won’t over-come. Until someone finds a way for applications to understand each other, the effect of Web services technology will be fairly limited. When I pass customer data across [the Web] in a certain format using a Web Services interface, the receiving program has to know what that format is. You have to agree on what the business objects look like. And no one has come up with a feasible way to work that out yet -- not Oracle, and not its competitors..."

Oracle Chairman and CEO Larry Ellison
The Vision

Deficiencies in Automated Information Processing
- finding
- extraction
- representation
- interpretation
- maintenance

Static

WWW
URI, HTML, HTTP

Semantic Web
RDF, RDF(S), OWL

Dynamic

Web Services
UDDI, WSDL, SOAP

Enable Computing over the Web

Static

WWW
URI, HTML, HTTP

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

Automated Web Service Usage

Dynamic

Web Services
UDDI, WSDL, SOAP

Semantic Web Services

Static

WWW
URI, HTML, HTTP

Semantic Web
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
- **Instance**
  - individual in the domain

Instance:
- Ann memberOf student
  - name = Ann Lee
  - studentID = 12345

Axiom:
- holds(Professor, Lecture) => Lecture.topic = Professor.researchField

Diagram:

- Person
  - Professor
  - Student
  - Lecture
  - holds
  - attends
  - topic

Annotation:
- Person
  - name
  - email
  - researchField
- Professor
  - name
  - researchField
- Student
  - name
  - email
- Lecture
  - lectureNo
  - topic

Semantic Web Services, AAAI 2006, Boston (MA), 17 July 2006
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:**
  - (collaborative) 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, Find Service
- **WSDL**: Points to Service, Describes Service
- **Web Service**: Communicates with XML Messages
- **Service Consumer**: SOAP

---

**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)
SOAP

- Simple Object Access Protocol
- W3C Recommendation

XML data transport:
- sender / receiver
- protocol binding
- communication aspects
- content

UDDI

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

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

=> 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 Frameworks

*Michael Stollberg*
Aims and Requirements

• Frameworks for Semantic Web Services need to
  – cover all aspects relevant for enabling automated Web service usage
  – define conceptual model & axiomatization (= semantics)
  – provide formal language for semantic descriptions

• Approaches (W3C Member Submissions)
  1. WSMO: Ontologies, Goals, Web Services, Mediators
  2. OWL-S WS Description Ontology (Profile, Service Model, Grounding)
  3. SWSF Process-based Description Model & Language for WS
  4. WSDL-S semantic annotation of WSDL descriptions

Web Service Modeling Ontology WSMO

• Comprehensive Framework for SESA
  Semantically Empowered Service-Oriented Architecture
  – 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
WSMO Working Groups

- Conceptual Model & Axiomatization for SWS
- Formal Language for WSMO
- Ontology & 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

W3C submission 13 April 2005
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

- 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)

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 Conceptual Syntax

wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

namespace { _"http://www.example.org/example#", dc _"http://purl.org/dc/elements/1.1/"}

ontology _"http://www.example.org/exampleOntology"
concept ID
  attr1 ofType A
  [...]

goal _"http://www.example.org/exampleGoal"
  [...]

webService _"http://www.example.org/exampleWS"
  [...]

WSML Logical Expressions

• Frame- and FOL based concrete syntax

• Elements:
  – Function symbols (e.g. $f()$)
  – Molecules (e.g. Human subClassOf Animal, John memberOf Human, John[name hasValue "John Smith"]).
  – Predicates (e.g. distance(?x,?y,?z))
  – Logical connectives (or, and, not, implies, equivalent, impliedBy, forall, exists)

• Example:

  ?x memberOf Human equivalent
  ?x memberOf Animal
  and ?x memberOf LegalAgent.
WSML Logical Expressions

wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

namespace { _"http://www.example.org/example#", dc _"http://purl.org/dc/elements/1.1/"

ontology _"http://www.example.org/exampleOntology"
concept ID
attr1 ofType A
[...]

goal _"http://www.example.org/exampleGoal"
[...]

webService _"http://www.example.org/exampleWS"
[...]

OWL and WSML

WSML aims at overcoming deficiencies of OWL
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

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

Connectors between components with mediation facilities for handling heterogeneities

WSMO Web Service Description

- complete item description
- quality aspects
- Web Service Management

Non-functional Properties
DC + QoS + Version + financial

Capability
functional 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’

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

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
- 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
- WSMO capability precondition

```owl
capability VTA:capability
sharedVariables {?item, ?passenger, ?creditCard, ?initialBalance, reservationPrice}
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.

```rml
assumption
  definedBy
    po#validCreditCard(?creditCard) and
    ?creditCard[balance hasValue ?initialBalance] and
    (?initialBalance >= ?reservationPrice).
```

Example VTA Web Service

• capability description (post-state)

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

effect
  definedBy
    ?creditCard[po#balance hasValue ?finalBalance] and
    (?finalBalance = (?initialBalance - ?reservationPrice)).
```
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

interface 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
- Web 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
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 $\text{GT}(\omega)$:
  – state transition
  – general structure: if (condition) then (update)
    • condition on current state, update = changes in state transition
    • all $\text{GT}(\omega)$ whose condition is fulfilled fire in parallel

Example Hotel Web Service

• choreography interface (state signature)

```xml
interface htl#BookHotelInterface

choreography

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
            endForall
        add(htl#notAvailable(?req)[trv#date hasValue ?date,
            trv#location hasValue ?loc] memberOf htl#HotelNotAvailable)
    ctl_state := htl#noAvail
endif
```

WSMO Goals

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)
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

- defines

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

Service-Side

- service detection & composition

(Web) Service Implementation

(Not of interest here)

Ontology

Domain Knowledge

Ontology
**Goal Model (WSMO 2.0)**

- **Abstract Goal**
  - importsOntology: ontology
  - precondition: axiom
  - effect axiom
  - resolutionConstraints: axiom

- **Atomic Goal**
  - nonFunctionalProperties: nFP
  - input: axiom
  - output: axiom
  - clientInterface: Interface

- **Composite Goal**
  - nonFunctionalProperties: nFP
  - input: axiom
  - output: axiom
  - subGoal: [atomic goal, composite goal]
  - controlDataFlow: controlDataFlow

**WSMO Mediators**

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
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 *Functionals*
  3. **Protocol & Process Level**: heterogeneous *Communication Processes*

**WSMO Mediators Overview**

Legend

- technique used
- imports / reuses
- correlation
Mediator Usage

Other Approaches

- WSMO is not the only proposal for an SWS Framework …

  **OWL-S:**
  - upper ontology for semantically describing Web services
  - chronologically first, consortium mainly USA

  **SWSF:**
  - process model for Web Services
  - result of SWSI (international working group)

  **WSDL-S:**
  - semantic annotation of WSDL descriptions
  - LSDIS Lap (Amit Seth Group) and IBM

- Discussed here:
  - Central Features
  - Commonalities and Differences
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 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:

**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
SWSF

• 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</td>
<td>ServiceAtomicProcess</td>
</tr>
<tr>
<td></td>
<td>activities</td>
<td>composedOf</td>
</tr>
<tr>
<td>Control Constraints</td>
<td>common workflow-style process constructs, enhancing OWL-S</td>
<td>SequenceOrderedChoice</td>
</tr>
<tr>
<td></td>
<td>process model concepts</td>
<td></td>
</tr>
<tr>
<td>Ordering Constraints</td>
<td>allow specification of activities defined by</td>
<td>OrderActivity</td>
</tr>
<tr>
<td></td>
<td>sequencing properties of atomic processes</td>
<td></td>
</tr>
<tr>
<td>Occurrence</td>
<td>support for nondeterministic activities within services</td>
<td>OccActivity</td>
</tr>
<tr>
<td>Constraints</td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Constraints</td>
<td>specify activities that are triggered by states (of an</td>
<td>TriggeredActivity</td>
</tr>
<tr>
<td></td>
<td>overall system)</td>
<td></td>
</tr>
<tr>
<td>Exception Constraints</td>
<td>basic infrastructure for modeling exceptions</td>
<td>Exception</td>
</tr>
</tbody>
</table>

WSDL-S

Semantic annotation of WSDL descriptions

1. annotate XML Schema with domain ontology

```xml
<xs:element name="processPOResponse" type="xs:string
wssem:modelReference="F0Ontology#OrderConfirmation"/>
```

2. pre-conditions & effects for operations

```xml
<interface name="PurchaseOrder">
  <operation name="processPurchaseOrder" pattern="in-out">
    <input messageLabel="processPORequest" element="tns:processPORequest"/>
    <output messageLabel="processPOResponse" element="processPOResponse"/>
    <wssem:precondition name="AccExitsPrecond" 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"/>
```
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

Summary

<table>
<thead>
<tr>
<th></th>
<th>OWL-S</th>
<th>WSMO</th>
<th>SWSF</th>
<th>WSDL-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope</td>
<td>description model for semantically describing Web services</td>
<td>description model &amp; language for core elements of Semantic Web service technologies</td>
<td>extension of the OWL-S Process Model</td>
<td>semantic annotation of WSDL descriptions</td>
</tr>
<tr>
<td>Top Level Elements</td>
<td>Service Profile, Process Model, Grounding</td>
<td>Ontologies, Goals, Web Services, Metamodels</td>
<td>Processes</td>
<td>Operations / WSDL descriptions</td>
</tr>
<tr>
<td>Service Level Description</td>
<td>non-functional aspects: ROPE for service-level functional description</td>
<td>capability (PAPE) for provided and requested functionality</td>
<td>not in scope</td>
<td>keyword classification (ontology-based)</td>
</tr>
<tr>
<td>Operation Level Description</td>
<td>ROPE for processes</td>
<td>interfaces for consumption (orchestration) and interaction (orchestration)</td>
<td>internal behavior (atomic and composite processes)</td>
<td>preconditions &amp; effects for WSDL operations</td>
</tr>
<tr>
<td>Language (static)</td>
<td>OWL</td>
<td>WSML</td>
<td>SWXML &amp; SWRUL</td>
<td>not specified</td>
</tr>
<tr>
<td>Language (dynamic)</td>
<td>Process Model and OWL</td>
<td>Abstract State Machines</td>
<td>FLOWS</td>
<td>not specified</td>
</tr>
</tbody>
</table>
PART III:

Semantic Techniques for Automated Web Service Usage

Michael Stollberg

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
Web Service Usage Process

Web Service Discovery

detect directly usable Web services out of available ones

- Discovery 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

- Selection: choose most appropriate Web Service with respect to:
  - Quality of Service (security, robustness, availability)
  - context (regional, business / social communities)
  - preferences and policies
  - financial
  - …
**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:**
\[ G, WS, O, M \models \neg \exists x. (G(x) \land WS(x)) \]


**Discovery Procedure**

- **Goal Capability**
  - **Postcondition**
  - **Effect**
  - **Precondition**
  - **Assumption**

- **Web Service Capability**
  - **Postcondition**
  - **Effect**
  - **Precondition**
  - **Assumption**

- \( \text{post} \Rightarrow \text{pre} \land \text{assumption} \land \text{effect} \)

- **Remarks:**
  - Precondition & assumption / postcondition & effect semantically the same
  - Only situation that guarantees goal resolution by Web service usage is
    \( \text{subsume} \iff \text{post} \land \text{plugin} \iff \text{pre} \land \text{assumption} \land \text{effect} \)

Semantic Web Services, AAAI 2006, Boston (MA), 17 July 2006
Web Service Composition

**combine several Web services for solving a request**

- need for composition
  - if no directly usable Web service exists …
    - a) a WS can satisfy goal, but goal cannot invoke WS
    - b) several WS need to be combined in order to achieve goal

- Types of Composition Techniques:
  - functional = suitable composition wrt **functionalities**
  - behavioral = suitable composition wrt **behavioral interfaces**

  ⇒ need to be integrated:
  1. skeleton by functional composition
  2. refinement + executable code by behavioral composition

---

**Procedure:**

1. directly usable WS (discovery)?
   - yes
   - no

2. composition (functional)?
   - yes
   - no
     - a) no
     - b) abort

3. composition (behavioral)?
   - yes
   - no
     - abort

4. executable composition

---

Functional Composition

**find suitable sequence of Web services for solving a goal with respect to functionality**

- mainly AI Planning on functional descriptions

\[
\begin{align*}
(q) & \quad \text{compCandidate}(Q, \{S\}) \leftarrow \exists S_i \in \{S_i, \ldots, S_n\}, \psi^{\text{pre}}_Q \equiv \psi^{\text{post}}_Q \land \psi^{\text{post}}_Q \equiv \psi^{\text{pre}}_Q \\
(k) & \quad \exists S_i \in \{S_i, \ldots, S_n\}, \psi^{\text{post}}_Q \equiv \psi^{\text{post}}_Q \land \psi^{\text{pre}}_Q \equiv \psi^{\text{pre}}_Q
\end{align*}
\]

for each \( S_i \in \text{compCandidate}(Q, \{S\}) \) do

1. add \( S_i \); \text{composition}(Q, S_i \circ S \circ \ldots \circ S)

2. create new request \( Q' : \phi^{\text{pre}}_Q = \phi^{\text{pre}}_Q \) and \( \phi^{\text{post}}_Q \equiv \phi^{\text{post}}_Q \land \phi^{\text{pre}}_Q 

3. detect candidates for \( Q' : \text{compCandidate}(Q', \{S\}) \)

repeat until \( \phi^{\text{post}}_Q \equiv \phi^{\text{pre}}_Q \).

- main technique: AI Planning
  - set of Web services with control & data flow
  - composition skeleton with all needed Web services
**Functional Composition Example**

**compose Flight & Hotel Booking Web service**

- **Goal:** book flight and hotel
  - $\Sigma_{b} = \{ \text{src, dest, fl., pl.} \}$
  - $\text{pre} = \text{memberOf(src, origin)} \land \text{memberOf(dest, destination)} \land$
    $\text{memberOf(fl., datetime)} \land \text{memberOf(pl., person)}$.
  - $\text{eff} = \text{src} \land \text{memberOf(src, purchaseContract)} \land$ $\text{fl.} \land \text{flightdate(dest, fl.)} \land \text{for(pl, fl.)} \land$
    $\text{memberOf(fl., purchaseContract)} \land \text{for(pl, hotel)} \land$
    $\text{in(dest, hotel)} \land \text{checkin(date, hotel)} \land \text{guest(pl, hotel)}$.

- **Situation (b):** flight & hotel WS need to be combined

  - compCandidate = (Flight WS, Hotel WS)

- **only 1 iteration of algorithm**
  - flight + hotel satisfy goal post-state
  - goal satisfies preconditions for both WS

- **control & data flow:**
  - dest.flight = loc.hotel
  - dt.flight = checkin.hotel

  $\Rightarrow$ 1. flight, 2. hotel + interleaved execution necessary

**Behavioral Composition**

- **does there exists an executable sequence for interaction wrt communication behavior of composed Web services?**
  - analyze behavioral interfaces
  - determine existence of valid choreography

  "**Choreography Discovery**"

- **techniques:**
  - model-checking (for state-based descriptions)
  - conformance testing (for process-based)

- **exponential time**
Choreography Discovery

**Goal**

- both choreography interfaces given ("static")
- correct & complete consumption of VTA

=> existence of a valid choreography?

- VTA Orchestration & Chor. Interfaces of aggregated WS given

=> existence of a valid choreography between VTA and each aggregated WS?

- 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

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

- if \( \emptyset \) then \( \text{request} \)
  \[ \Omega_G(\omega 1) = \{ \text{request}(\text{out}) \} \]
- if \( \text{cnd1}(\text{offer}) \) then \( \text{changeReq} \)
  \[ \Omega_G(\omega 2a) = \{ \text{offer}(\text{in}), \text{changeReq}(\text{out}) \} \]
- if \( \text{cnd2}(\text{offer}) \) then \( \text{order} \)
  \[ \Omega_G(\omega 2b) = \{ \text{offer}(\text{in}), \text{order}(\text{out}) \} \]
- if \( \text{conf} \) then \( \emptyset \)
  \[ \Omega_G(\omega 3) = \{ \text{offer}(\text{in}), \text{conf}(\text{in}) \} \]

Goal Behavior Interface

VTA Behavior Interface

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

- if \( \text{request} \) then \( \text{offer} \)
  \[ \Omega_{VTA}(\omega 1) = \{ \text{request}(\text{in}), \text{offer}(\text{out}) \} \]
- if \( \text{changeReq} \) then \( \text{offer} \)
  \[ \Omega_{VTA}(\omega 2a) = \{ \text{changeReq}(\text{in}), \text{offer}(\text{out}) \} \]
- if \( \text{order} \) then \( \text{conf} \)
  \[ \Omega_{VTA}(\omega 2b) = \{ \text{order}(\text{in}), \text{conf}(\text{out}) \} \]

valid choreography existent

Orchestration Validation Example

VTA Web Service Orchestration

- if \( \emptyset \) then \((\text{FWS, flightRequest})\)
- if \( \text{flightOffer} \) then \((\text{HWS, hotelRequest})\)
- if \( \text{selection} \) then \((\text{FWS, flightBookingOrder})\)
- if \( \text{selection, flightBookingConf} \) then \((\text{HWS, hotelBookingOrder})\)

Flight WS Behavior Interface

- if \( \text{request} \) then \( \text{offer} \)
  \[ \text{Start (VTA, FWS)} \]
- if \( \text{order} \) then \( \text{confirmation} \)
  \[ \text{Termination (VTA, FWS)} \]

Hotel WS Behavior Interface

- if \( \text{request} \) then \( \text{offer} \)
  \[ \text{Start (VTA, HWS)} \]
- if \( \text{order} \) then \( \text{confirmation} \)
  \[ \text{Termination (VTA, HWS)} \]

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 functionalities
  - 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, 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
Functional Level Mediation

- requested and provided functionalities do not match precisely
- delta-relations = relation & difference of functional descriptions

<table>
<thead>
<tr>
<th>Subsumption</th>
<th>$\phi \models \psi$, i.e.: $\forall x. \phi(x) \Rightarrow \psi(x)$.</th>
<th>$\Delta = \phi \land \neg \psi$.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\phi = \psi \lor \Delta$.</td>
<td>$\psi = \phi \land \neg \Delta$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Plugin</th>
<th>$\phi \models \psi$, i.e.: $\forall x. \phi(x) \Rightarrow \psi(x)$.</th>
<th>$\Delta = \phi \land \neg \phi$.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\phi = \psi \land \neg \Delta$.</td>
<td>$\psi = \phi \lor \Delta$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intersection</th>
<th>$\exists x. (\phi(x) \land \psi(x)) \land \neg (\forall x. \phi(x) \Rightarrow \psi(x))$.</th>
<th>$\Delta = (\phi \land \neg \phi) \lor (\neg \phi \land \psi)$.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\phi = \phi \land \Delta$, $\psi = \phi \land \Delta$.</td>
<td>$\psi = \psi \lor \Delta$.</td>
</tr>
</tbody>
</table>

- Beneficial Usage
  - goal refinement
  - goal adjustment
  - grouping functionalities (goals and Web services) for efficient search

Protocol Level Mediation

- interoperability problems due to
  - different representation formalisms
  - different technical communication protocols
- adaptors for transformation
  - syntactic transformation
  - mappings between language constructs
- usage:
  - interoperability between systems with different languages (e.g. OWL – WSML, etc.)
  - grounding for Semantic Web services (lifting & lowering between syntactic and semantic level)
Process Level Mediation

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

- partially resolvable by “process mediation patterns”

Summary

- techniques for automated Web service usage apply results from various AI disciplines
  - Knowledge Representation
  - Formal Software Reuse
  - AI Planning
  - Business Process & Workflow Engineering
  - Data Integration
  - Web technologies
  - ...

- Status of Development
  - first set of solutions with converging techniques
  - integration & automated combination as next step
PART IV:
Standardization
Market Prospect
Future Issues

Michael Stollberg

History I

- late 90s: TBL wants the Internet to develop further
  - HTML is unstructured => not processable by machines
  - New kinds of Web Technologies needed
    => “turn the internet from a world-wide information repository for human consumption into a device of world-wide distributed computation” (Fensel & Bussler, WSMF)

- American Scientific Article “The Semantic Web”
  - Pete & Lucy: a future example
  - Core Technologies:
    - Ontologies: unambiguous terminology definition in machine-readable format (“Semantics”)
    - Web Services: functionality evocable over the Internet, re-usable and combinable distributed software components
    - Agents: electronic representatives that perform tasks on behalf of his owner

- Rising attention in Research & Industry ..
History II

- **1999: first W3C Recommendations**
  - Specifications of XML Technologies (XSL, XSLT, …)
  - Semantic Web Layer Cake
  - Languages: XML, RDF

- **2000 – 2001: first R&D-activities**
  - 1. Web Service Technology Specifications: SOAP, WSDL, UDDI
  - related research areas become interested (AI / Knowledge Engineering; distributed computing, etc.), first projects: DAML (US), OntoKnowledge, etc.
  - “1st Semantic Web Working Symposium”, Stanford (USA), ca. 100 participants

- **2002 – 2003: research & industry sets off**
  - SDK-Cluster (Europe), DAML efforts (USA)
  - initial research results, still very chaotic / without a “framework”
  - industrial efforts on Web services
  - ISWC 02 / 03: double number of participants each year

- **2004 ff: the hot phase**
  - W3C recommendations (OWL, XML + RDF revisions, others)
  - first set of research & development results
  - rising industrial & commercial attention

Standardization Efforts W3C

- **1st set of recommendations in 1999 / 2000, currently revised**

<table>
<thead>
<tr>
<th>Name</th>
<th>Topic</th>
<th>Status</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML 1.1</td>
<td>XML Spec revision</td>
<td>Recommendation</td>
<td>04 Feb 2004</td>
</tr>
<tr>
<td>Semantic Web</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework, revision</td>
<td>Recommendation</td>
<td>10 Feb 2004</td>
</tr>
<tr>
<td>OWL</td>
<td>ontology language for the Semantic Web</td>
<td>Recommendation</td>
<td>10 Feb 2004</td>
</tr>
<tr>
<td>SPARQL</td>
<td>RDF query language</td>
<td>Candidate Recommendation</td>
<td>06 Apr 2006</td>
</tr>
<tr>
<td>Web Services</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOAP 1.2</td>
<td>XML Messaging</td>
<td>Recommendation</td>
<td>24 June 2003</td>
</tr>
<tr>
<td>WSDL 2.0</td>
<td>Web service description</td>
<td>Candidate Recommendation</td>
<td>27 March 2006</td>
</tr>
<tr>
<td>WS Addressing</td>
<td>endpoint message referencing</td>
<td>Proposed Recommendation</td>
<td>21 March 2006</td>
</tr>
<tr>
<td>WS-CDL</td>
<td>choreography description language</td>
<td>Candidate Recommendation</td>
<td>09 Nov 2005</td>
</tr>
</tbody>
</table>

- **Semantic Web Services**
  - Member Submissions: OWL-S, WSMO, SWSF, WSDL-S
  - Working Groups:
    - Semantic Web Service Interest Group
    - Semantic Annotations for WSDL Group

=> *standardization need acknowledged, but no agreement yet on what & how*
Layer Cake - Revised

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

Industrial Efforts

- Semantics & SOA Developments
  - Microsoft  Longhorn / Vista / Biztalk Server 2006 / …
  - IBM  IBM SOA Foundation
  - SAP  Net Weaer
  - Oracle  Oracle SOA Suite
  - Sun  SOA Initiative (future developments)

- OASIS
  - non-profit, joint industrial for e-business technology development & standardization
  - committees for Web Services & SOA (ebSOA, FWSI, SEE, etc.)
Market Prospects

- Application Areas
  - Knowledge Management
  - Enterprise Application Integration
  - E-Commerce (B2C and B2B)
  - E-Government
  - … many more

**SESA = enabling technology for the 21st century**

- Market Prospects:
  - 2006 / 07: Technology Development & Dissemination
  - 2008: Break Even Point / ROI
  - 2010: Commercialization (40 – 60 billion dollar market)

Market Development (Gartner)

![Chart showing market development](chart.png)

- Standards Power B2B Commerce After 2008 ‘Sweet Spot’ is Reached
- The semantic Web emphasizes XML and machine-readable content
- Development and testing remaining on core Web services standards stack: SOAP, WSDL, UDDI, BPEL, WS-Events, others
- 2008 Sweet Spot: Growth of semantic B2B standards
- Boom in B2B Trading Efficiencies
- These Standards Are All About Meaning of Terms: OWL, RDF, UCC/RosettaNet, HL 7, Accord, others
Estimated Market in 2010

- $52.4 billion dollar market

Future Items

1. proof of concept & applicability
   - current works developed & tested in mainly academic settings
   - which approaches techniques are
     • adequate (functional, scalable, etc.)
     • realizable
   ⇒ large scale real world use cases needed

2. Ontology & WS description management
   - Ontologies as data model
     ⇒ the (Web) world needs to be ontologized
   - Web service descriptions must be correct & maintained
     • complicated task
     • can not be automated (knowledge level lifting)
   ⇒ qualified Knowledge Engineers needed
PART IV:

The Web Service Execution Environment

WSMX

Emilia Cimpian

WSMX Introduction

• Software framework for runtime binding of service requesters and service providers
• WSMX interprets service requester’s goal to
  – discover matching services
  – select (if desired) the service that best fits
  – provide mediation (if required)
  – make the service invocation
• Is based on the conceptual model provided by WSMO
• Has a formal execution semantics
• Service Oriented and event-based architecture
  – based on microkernel design using technologies as J2EE, Hibernate, Spring, JMX, etc.
WSMX Motivation

- Middleware ‘glue’ for Semantic Web Services
  - Allow service providers focus on their business
- Reference implementation for WSMO
  - Eat our own cake
- Environment for goal based 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 Scenario
WSMX Usage Scenario - P2P

- A P2P network of WSMX 'nodes'
- Each WSMX node described as a SWS
- Communication via WSML over SOAP
- Distributed discovery – first aim
- Longer term aim - distributed execution environment
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
Benefits of SOA

• Better reuse
  – Build new functionality (new execution semantics) on top of existing Business Services

• Well defined interfaces
  – Manage changes without affecting the Core System

• Easier Maintainability
  – Changes/Versions are not all-or-nothing

• Better Flexibility

Service Oriented State

• The interface to the service is implementation-independent

• The service can be dynamically invoked
  – Runtime binding

• The service is self-contained
  – Maintains its own state
Messaging

- Messaging is peer-to-peer facility
- Distributed communication
  - Loosely coupled
- Sender does not need to know receiver (and vice versa)
- Asynchronous mechanism to communicate between software applications

![WSMX Architecture Diagram](image-url)
Selected Components

- Adapters
- Parser
- Invoker
- Choreography
- Process Mediator
- Discovery
- Data Mediator
- Resource Manager
- Reasoning

Adapters

- To overcome data representation mismatches on the communication layer
- Transforms the format of a received message into WSML compliant format
- Based on mapping rules
**Parser**

- WSML compliant parser
  - Code handed over to wsmo4j initiative
- Validates WSML description files
- Compiles WSML description into internal memory model
- Stores WSML description persistently (using Resource Manager)

**Communication Mgr – Invoker**

- WSMX uses
  - The SOAP implementation from Apache AXIS
  - The Apache Web Service Invocation Framework (WSIF)
- WSMO service descriptions are grounded to WSDL
- Both RPC and Document style invocations possible
- Input parameters for the Web Services are translated from WSML to XML using an additional XML Converter component.
Choreography

- Requester and provider have their own observable communication patterns
  - Choreography part of WSMO
- Choreography instances are loaded for the requester and provider
  - Both requester and provider have their own WSMO descriptions
- Choreography Engine
  - Evaluation of transition rules - prepares the available data
  - Sends data to the Process Mediator - filters, changes or replaces data
  - Receives data from PM and forwards it to the Communication manager - data to be finally sent to the communication partner

Process Mediator

- Requester and provider have their own communication patterns
- Only if the two match precisely, a direct communication may take place
- At design time equivalences between the choreographies’ conceptual descriptions is determined and stored as set of rules
- The Process Mediator provides the means for runtime analyses of two choreography instances and uses mediators to compensate possible mismatches
Process Mediator – Addressed mismatches

Process Mediator – Unsolvable Mismatches
Discovery

• Responsible for finding appropriate Web Services to achieve a goal (discovery)
• Current discovery component is based on simple matching
  – Keywords identified in the NFP of the goal
  – Matched against NFPs of the published WSs
  – Variable set of NFPs to be considered for this process
  – To be extended
    • Values in NFPs might be concepts from ontologies
    • More elaborate string matching algorithms

• Advanced semantic discovery in prototypical stage
Data Mediator

• Ontology-to-ontology mediation
• A set of mapping rules are defined and then executed
• Initially rules are defined semi-automatic
• Create for each source instance the target instance(s)

Design-time

• Inputs
  – Source Ontology and Target Ontology
• Features
  – Graphical interface
  – Set of mechanism towards semi-automatic creation of mappings
  – Capturing the semantic relationships identified in the process
  – Storing these mappings in a persistent storage
• Output
  – Abstract representation of the mappings
Design-time Phase

Design-time Phase - Approach, Decomposition and Mapping Context

- Bottom-up -> training set
- Top-down -> decomposition, context
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 (ex: classMapping, classAttributeMapping, instanceMapping)
  - mapping operators

Mapping Language Example

```
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
Run-Time Data Mediator

- Main Mediation Scenario: Instance Transformation

- Inputs
  - Incoming data
    - Source ontology instances

- Features
  - Completely automatic process
  - Grounding of the abstract mappings to a concrete language
    - WSML
  - Uses a reasoner to evaluate the mapping rules
    - MINS

- Outputs
  - Mediated data
    - Target ontology instances

Resource Manager

- Stores internal memory model to a data store
- Decouples storage mechanism from the rest of WSMX
- Data model is compliant to WSMO API
- Independent of any specific data store implementation i.e. database and storage mechanism
Reasoner

- **Mins**
  - Datalog + Negation + Function Symbols Reasoner Engine
  - Features
    - Built-in predicates
    - Function symbols
    - Stratified negation

- **WSMO4J**
  - Validation, serialization and parsing

- **WSML2Reasoner**
  - Reasoning API
    - Contains:
      - Common API for WSML Reasoners
      - Transformations of WSML to tool-specific input data (query answering or instance retrieval)

- **WSML-DL-Reasoner**
  - Features:
    - T-Box reasoning (provided by FaCT++)
    - Querying for the equivalents, for the children, for the descendants, for the parents and for all ancestors of a given concept
    - Testing the satisfiability of a given concept with respect to the knowledge base
    - Subsumption test of two concepts with respect to the knowledge base
    - Wrapper of WSML-DL to the XML syntax of DL used in the DIG interface

System Entry Points

- **achieveGoal** (WSMLDocument): Context
- **getWebServices** (WSMLDocument): Context
- **invokeWebService**(WSMLDocument, Context): Context
Define “Business” Process

1. Define Business Process
2. Generate Wrappers for Components
Context Data

Event-based Implementation
Conclusions

- Conceptual model is WSMO
- End to end functionality for executing SWS
- Has a formal execution semantics
- Real implementation
- Open source code base at SourceForge
- Event-driven component architecture

PART IV:

The Internet Reasoning Service - IRS III
and
WSMO Studio

John Domingue
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 Framework

IRS-3 Server
- Domain Models
- Web Service Specifications + Registry of Implementors
- Goal Specifications + SOAP Binding

IRS Publisher
- Lisp
- Java
- Java WS

IRS Client

IRS-III Architecture

Publishing Platforms
- Web Service
- Java Code
- Web Application

SOAP
- Browser Handler
- Publisher Handler
- Invocation Handler

LispWeb Server
- WSMX
- WSMO Studio
- Browser
- Publishing Clients
- Invocation Client

Java
- WSMO
- WSML Studio
- OCML
- WSMO Library
Publishing Platform Architecture

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)
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-mediated description
  - Source = goal
  - Possibly add a mediation service
- Create a web service description
  - Used-mediated of WS capability = wg-mediated 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-mediated 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

- **Web service selection**
  European-exchange-rate

- **Mediation**
  Mediate input values
  ‘$’ -> us-dollar
  European-exchange-rate

- **Invocation**
  Invoke selected web service
  European-exchange-rate
WSMO Studio

• Integrated Service Environment for WSMO
• Provide easy to use GUI for various WSMO tasks
  – Working with ontologies
  – Creating WSMO descriptions: goals, services, mediators
  – Creating WSMO centric orchestration and choreography specifications
  – Import (export) from (to) various formats
  – Front-end for ontology and service repositories
  – Front-end for runtime SWS environments (WSMX, IRS-III)
• http://www.wsmostudio.org

Requirements for an ISE

• Modular design
  – Different users need to customise the functionality in a specific way
  – Easier to maintain (e.g. ship new versions and bugfixes)
  – More suitable for 3rd party contributions
• Extensibility
  – SWS is an emerging domain
  – It is difficult to specify requirements and functionality afront
• Architecture based on open standards
  – Lowers the cost of adopting / integrating a tool
  – 3rd party extensions and improvements are more likely to occur
• Flexible licensing
  – An Open Source licence improves the adoption rate
WSMO Studio

- Java based implementation
- Open Source core
  - LGPL
  - 3rd party contributors are free to choose their respective licensing terms
- Modular design
  - an Eclipse based plug-in architecture
- Extensible
  - 3rd parties may contribute new functionality (plug-ins) or modify existing functionality

Editing a Goal in WSMO Studio
PART VI:

Hands-On Session
(with IRS III)

John Domingue
Liliana Cabral
European Travel Scenario

European Travel Demo
Goal Description

• Goals describe requirements from client perspective…

Semantic Web Services, AAAI 2006, Boston (MA), 17 July 2006

Goal Description

• Their Capabilities describe the functional requirements…

Semantic Web Services, AAAI 2006, Boston (MA), 17 July 2006
Goal Description

- Preconditions express guarantees client can make, purely over information they can communicate, in order that functional requirements are met...

- Assumptions express general guarantees client can make, involving communications and environment, in order that functional requirements are met...
Goal Description

- Postconditions express guarantees client would like over information communicated back in order that functional requirements are met...

- Effects express the general guarantees the client would like after the goal has been achieved.
Goal Description

- Capabilities can be used for one or more of: representing a client-oriented perspective, advertising and service discovery. We do not use goal capabilities in the hands on session.

- The interfaces of goals describe the behavioural requirements of clients, i.e. constraints over communication
• The choreography expresses communications the client is able to engage in...

• The state signature describes these communications semantically, by linking modes to ontological concepts
The state signature describes these communications semantically, by linking modes to ontological concepts:

- **IN** modes describe communications the client would like to receive;
- **OUT** modes describe communications the client is able to send.
• Transition rules link communications into a stateful interaction:
  – Transition rules can be used to constrain the stateful behaviour of matching
    services, or define the process mediation 'a priori'. We do not use transition rules
    in the hands on session.
Goal Description

- Orchestrations govern over the composite behaviour that is required to go into meeting the goal – the technology to exploit this is not yet available.

Goal Description in Tutorial

- The steps that go into describing a goal in the tutorial are:
The steps that go into describing a goal in the tutorial are:
- Ontological description of the communications (request and response);
- Creation of a goal
• The steps that go into describing a goal in the tutorial are:
  – Ontological description of the communications (request and response);
  – Creation of a goal;
  – Attachment of a choreography;
  – Attachment of a state signature.
Goal Description in Tutorial

Goal
Capability
Precondition
Assumption
Postcondition
Effect

Interface
Choreography
State Signature
Transition Rules
Orchestration

State Signature
Transition Rules

• The steps that go into describing a goal in the tutorial are:
  – Ontological description of the communications (request and response);
  – Creation of a goal;
  – Attachment of a choreography; Attachment of a state signature;
  – Attachment of communications to state signature:
    • request as OUT mode; response as IN
WSMO Web Services describe abilities of deployed services…

Their Capabilities describe their functional abilities…
Web Service Description

- Preconditions express guarantees they expect from clients, purely over information they communicate...

- Assumptions express general guarantees they expect of clients, involving communications and environment...
• Postconditions express guarantees they make over information communicated back, providing the preconditions and assumptions are met by the client...

• Effects express the general guarantees made, over communicated and changes to the environment, providing the preconditions and assumptions are met by the client...
The last part of the hands on session uses the assumption for web service selection.

The interfaces of web services describe their behavioural characteristics, *i.e.* the communications they engage in.
Web Service Description

Web Service
  Capability
    Precondition
    Assumption
    Postcondition
    Effect
  Interface
    Choreography
      Orchestration
      Mediation Goal
      OO-Mediator
    WG-Mediator
  State Signature
    Transition Rules

• The choreography expresses communications the service engages in with its clients...

Web Service Description

Web Service
  Capability
    Precondition
    Assumption
    Postcondition
    Effect
  Interface
    Choreography
      Orchestration
      Mediation Goal
      OO-Mediator
    WG-Mediator
  State Signature
    Transition Rules

• The state signature describes these communications semantically, by linking modes to ontological concepts
Web Service Description

• The state signature describes these communications semantically, by linking modes to ontological concepts:
  – IN modes describe communications the service is able to receive;
  – OUT modes describe communications the service is able to send.
Web Service Description

- **Capability**
  - Precondition
  - Assumption
  - Postcondition
  - Effect

- **Interface**
  - Choreography
  - Orchestration
  - Mediation Goal
  - OO-Mediator

- **State Signature**
  - Transition Rules

- **Choreography**
  - WG-Mediator

- **Orchestration**

- **State Signature**
  - Transition Rules

- **Interface**
  - WG-Mediator

- **Precondition**
  - Mediation Goal

- **Postcondition**
  - OO-Mediator

- **Effect**

- **Choreography**

- **Orchestration**

- **SOA Asynchronous Agent**

- **Mediation Goal**

- **OO-Mediator**

**Semantic Web Services, AAAI 2006, Boston (MA), 17 July 2006**

- The state signature describes these communications semantically, by linking modes to ontological concepts:
  - IN modes describe communications the client would like to receive;
  - OUT modes describe communications the service is able to send;
  - modes may be grounded to physical communications for service execution (SOAP endpoints, REST identifiers, LISP and Java functions).

- Transition rules link communications into a stateful interaction.
Transition rules link communications into a stateful interaction:
- Transition rules may be used in matching and (process) mediation against goals,
- In process mediation between IRS-III/WSMX broker and the deployed service
Web Service Description

- Orchestrations describe how composite services achieve their behaviour in terms of communications between its components, which may be goals or services. We do not cover this in the hands on session.

- WG-Mediators describe which goals are met by a web service.
Web Service Description

- WG-Mediators describe which goals are met by a web service;
- the descriptions may have some mismatch to be mediated:
  - a mediation goal describes data mediation which needs to take place between client communications and those of the service
Web Service Description

- WG-Mediators describe which goals are met by a web service;
- the descriptions may have some mismatch to be mediated:
  - a mediation goal describes data mediation which needs to take place between client communications and those of the service;
  - an oo-mediator can map between descriptions in two different ontologies – we do not cover this in the hands on session.

Web Service Description in Tutorial

- The steps that go into describing a service in the tutorial are:
Web Service Description in Tutorial

- The steps that go into describing a service in the tutorial are:
  - Ontological description of the communications (may be reused from goal)
  - Creation of a service
The steps that go into describing a service in the tutorial are:

- Ontological description of the communications (may be reused from goal);
- Creation of a service; possibly attachment of an assumption

The steps that go into describing a service in the tutorial are:

- Ontological description of the communications (may be reused from goal);
- Creation of a service; possibly attachment of an assumption
- Creation of a wg-mediator (possibly involving a mediation goal)
The steps that go into describing a service in the tutorial are:
- Ontological description of the communications (may be reused from goal);
- Creation of a service; possibly attachment of an assumption
- Creation of a wg-mediator (possibly involving a mediation goal);
- Attachment of a choreography

Semantic Web Services, AAAI 2006, Boston (MA), 17 July 2006
Web Service Description in Tutorial

- The steps that go into describing a service in the tutorial are:
  - Ontological description of the communications (may be reused from goal);
  - Creation of a service; possibly attachment of an assumption
  - Creation of a wg-mediator (possibly involving a mediation goal);
  - Attachment of a choreography; Attachment of a state signature;
  - Attachment of communications to state signature:
    - request as IN mode
Web Service Description in Tutorial

- Ontological description of the communications (may be reused from goal);
- Creation of a service; possibly attachment of an assumption
- Creation of a wg-mediator (possibly involving a mediation goal);
- Attachment of a choreography; Attachment of a state signature
- Attachment of communications to state signature:
  - request as IN mode, grounded to LISP function

Web Service

- Capability
  - precondition
  - assumption
  - postcondition
  - effect

- Interface
  - Choreography
    - Orchestration
  - Mediation Goal
    -OO-Mediator

- WG-Mediator
  - Mediation Goal
  - OO-Mediator
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.

- The following WSMO Studio tasks are involved:
  - Retrieve domain ontology from IRS;
  - Create WSML ontology concepts to describe communications;
  - Create WSMO descriptions for Goals, WG-mediators and Web service descriptions;
  - Export these definitions to the IRS;
  - Create WSML ontology instances of the requests;
  - Achieve the goals against these instances.

Tutorial Setup

```
 IRS Server (3000)    IRS Lisp Publisher    Travel Services (3001)
```

```
WSMO Studio
```

```
Domain Models
```
Travel Related Knowledge Models

Key Classes, Relations, Instances

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."
Available Functions (2/3)

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."

Available Functions (3/3)

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
Wrap Up
References
Acknowledgements

Tutorial Wrap-up

The targets of the presented tutorial were to:

- understand aims & challenges within Semantic Web Services
- understand WSMO and other frameworks
  - design principles & paradigms
  - core elements
  - commonalities and differences
- understand semantic techniques for automated Web service usage and give:
  - .. Semantic Web Service Tools and System Presentation
  - .. do-it-yourself Hands-On Session

=> you should now be able to assess technologies & products for Semantic Web Services and utilize these for your future work
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


References WSMO

- The central location where WSMO work and papers can be found is WSMO Working Group: [http://www.wsmo.org](http://www.wsmo.org)


- WSMO implementation
  - WSMX working group : [http://www.wsmx.org](http://www.wsmx.org)
  - WSMX open source can be found at: [https://sourceforge.net/projects/wsmx/](https://sourceforge.net/projects/wsmx/)
References IRS III


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

Acknowledgements

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.

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

Slide design by Roberta Hart-Hilber, DERI Austria