THE WEB SERVICES MODELLING ONTOLOGY
2005 OASIS SYMPOSIUM - NEW ORLEANS

CHRISTOPH BUSSLER
ADRIAN MOCAN
MATTHEW MORAN
MICHAEL STOLLBERG
MICHAL ZAREMBA
LILIANA CABRAL
JOHN DOMINGUE
# Table of Contents

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
<th>Presenter(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:30 – 3:00</td>
<td>Semantic Web Services</td>
<td>Michal Zaremba</td>
</tr>
<tr>
<td></td>
<td>Web Services Modelling Ontology</td>
<td>Adrian Mocan</td>
</tr>
<tr>
<td>3:00 - 3:30</td>
<td>Coffee break</td>
<td></td>
</tr>
<tr>
<td>3:30 - 4:10</td>
<td>WSMO Continue</td>
<td>Adrian Mocan</td>
</tr>
<tr>
<td></td>
<td>Web Services Modelling Language</td>
<td>Michal Zaremba</td>
</tr>
<tr>
<td></td>
<td>Web Services Execution Environment</td>
<td></td>
</tr>
<tr>
<td>4:10 - 5:00</td>
<td>IRSIII &amp; Demo</td>
<td>Liliana Cabral</td>
</tr>
<tr>
<td></td>
<td>WSMO Tools</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Summary, Conclusions &amp; Future Work</td>
<td></td>
</tr>
</tbody>
</table>
Semantic Web Services

Michal Zaremca
Semantic Web - The Vision

- 500 million users
- more than 3 billion pages

Dynamic

Static

WWW

URI, HTML, HTTP

Syntax

Semantics
Semantic Web - The Vision

Serious Problems in
- information finding,
- information extracting,
- information representing,
- information interpreting and
- and information maintaining.

Dynamic

Static

Syntax

Semantics

WWW
URI, HTML, HTTP

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

Dynamic
Web Services
UDDI, WSDL, SOAP

Static
WWW
URI, HTML, HTTP

Syntax
Semantic Web
RDF, RDF(S), OWL

Bringing the computer back as a device for computation
Semantic Web - The Vision

Bringing the web to its full potential

Dynamic

Static

Web Services
UDDI, WSDL, SOAP

Intelligent Web Services

Semantic Web
RDF, RDF(S), OWL

WWW
URI, HTML, HTTP

Syntax

Semantics
Ontology Definition

formal, explicit specification of a shared conceptualization

- unambiguous definition of all concepts, attributes and relationships
- machine-readability
- commonly accepted understanding
- conceptual model of a domain (ontological theory)
Ontology Example

**Concept**
conceptual entity of the domain

**Property**
attribute describing a concept

**Relation**
relationship between concepts or properties

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

```
holds(Professor, Lecture) :-
Lecture.topic ∈ Professor.researchField
```
Ontology Languages

• Requirements:
  – ”expressivity“
    • knowledge representation
    • ontology theory support
  – ”reasoning support“
    • sound (unambiguous, decidable)
    • support reasoners / inference engines

• Semantic Web languages:
  – web compatibility
  – Existing W3C Recommendations:
    • XML, RDF, OWL
“Semantic Web Language Layer Cake”
Web Services

Web Services: [Stencil Group]
- 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
Web Services Problems
Web Services Problems
Lack of SWS standards

Current technology does not allow realization of any of the parts of the Web Services’ usage process:

- Only syntactical standards available
- Lack of fully developed markup languages
- Lack of marked up content and services
- Lack of semantically enhanced repositories
- Lack of frameworks that facilitate discovery, composition and execution
- Lack of tools and platforms that allow to semantically enrich current Web content
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 data interpretation (Semantic Web aspect)

- Define semantically driven technologies for automation of the Web Service usage process (Web Service aspect)
Semantic Web Services (2)

Usage Process:
- **Publication**: Make available the description of the capability of a service
- **Discovery**: Locate different services suitable for a given task
- **Selection**: Choose the most appropriate services among the available ones
- **Composition**: Combine services to achieve a goal
- **Mediation**: Solve mismatches (data, protocol, process) among the combined
- **Execution**: Invoke services following programmatic conventions
Semantic Web Services (3)

Usage Process – 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
Conclusion

Semantic Web Services
= Semantic Web Technology + Web Service Technology
Web Service Modelling Ontology (WSMO)

Adrian Mocan
Features

• WSMO is a complete conceptual model for Semantic Web Services and related aspects

• Identifies four main elements: Web Services, Goals, Ontologies, and Mediators
Overview

- WSMO Working Groups
- WSMO Design Principles
- WSMO Top Level Notions
  - Ontologies
  - Goals
  - Web Services
  - Mediators
- Basic Notions of WSML
- Using WSMO to address Web Services problems
  - Discovery
  - Composition
  - Grounding
WSMO Working Groups

A Conceptual Model for SWS

A Formal Language for WSMO

A Rule-based Language for SWS

Execution Environment for WSMO
WSMO 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)

Execution Semantics
reference implementation (WSMX)
WSMO Top Level Notions

Objectives that a client may have when consulting a Web Service

Provide the 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

Ontologies

Goals

Web Services

Mediators
Non-Functional Properties

• Every WSMO elements is described by properties that contain relevant, non-functional aspects of the item
• Used for management and element overall description
• Core Properties:
  - Dublin Core Metadata Element Set plus version (evolution support)
  - W3C-recommendations for description type
• Web Service Specific Properties:
  - Quality aspects and other non-functional information of Web Services
  - Used for Service Selection
Non-Functional Properties

ontology <http://www.wsmo.org/2004/d3/d3.2/v0.1/20040628/dt.wsml>

nonFunctionalProperties

    dc:title    "Date and Time Ontology"
    dc:creator "DERI International"
    dc:subject "Date", "Time", "Date and Time Algebra"
    dc:description "generic representation of data and time including basic algebra"
    dc:publisher "DERI International"
    dc:contributor "Holger Lausen", "Axel Polleres", "Ruben Lara"
    dc:date     2004-06-28
    dc:type     http://www.wsmo.org/2004/d2/v0.3/20040329/#ontos
    dc:format   "text/plain"
    dc:language "en-US"
    dc:relation <http://www.w3.org/TR/xmlschema-2/>
    dc:coverage "World"
    dc:rights   <http://www.deri.org/privacy.html>
    version     1.21
WSMO Ontologies

Objectives that a client may have when consulting a Web Service

Provide the 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
Ontology Specification

- Non functional properties
- Imported Ontologies Importing existing ontologies where no heterogeneities arise
- Used mediators: OO Mediators (ontology import with terminology mismatch handling)
- ‘Standard’ Ontology Notions:
  - Concepts set of concepts that belong to the ontology
  - Attributes set of attributes that belong to a concept
  - Relations: define interrelations between several concepts
  - Functions: special type of relation (unary range = return value)
  - Instances: set of instances that belong to the represented ontology
  - Axioms axiomatic expressions in ontology (logical statement)
WSMO Goals

Objectives that a client may have when consulting a Web Service

Provide the formally specified terminology of the information used by all other components

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

Connectors between components with mediation facilities for handling heterogeneities
Goals

- **De-coupling of Request and Service**
  
  **Goal-driven Approach**, derived from AI rational agent approach
  
  - Requester formulates objective independent / without regard to services for resolution
  
  - ‘Intelligent’ mechanisms detect suitable services for solving the Goal
  
  - Allows re-use of Services for different purposes

- **Usage of Goals within Semantic Web Services**
  
  - A Requester, that is an agent (human or machine), defines a Goal to be resolved
  
  - Web Service Discovery detects suitable Web Services for solving the Goal automatically
  
  - Goal Resolution Management is realized in implementations
Goal Specification

- Non functional properties
- Imported Ontologies
- Used mediators
  - *OO Mediators:* - import ontologies with integration
  - *GG Mediators:* - allow goal definition by reusing an already existing goal
    - allow specification of *Goal Ontologies*

- **Post-conditions** - the state of the information space that is desired.
  - The result expected from execution a Web Service
  - Expressed as an axiom (unambiguous, based on ontology)

- **Effects** - the state of the world that is desired.
  - Expected changes in the world that should hold after a service execution
  - Expressed as an axiom (unambiguous, based on ontology)
Objectives that a client may have when consulting a Web Service

Provide the formally specified terminology of the information used by all other components

Connectors between components with mediation facilities for handling heterogeneities

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

- Complete item description
- Quality aspects
- WS Management

Non-functional Properties

- Advertise of Web Service
- Support for WS Discovery

Capability

Functional description

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

Realization of WS by using other WS
- Functional decomposition
- WS Composition

Choreography --- Interfaces --- Orchestration

Interaction Interface
for consuming WS
- Messages
- External Visible Behavior
- Grounding

Core + WS-specific
Web Service specific Properties

- Non-functional information of Web Services:
  - Accuracy
  - Availability
  - Financial
  - Network-related QoS
  - Performance
  - Reliability
  - Robustness
  - Scalability
  - Security
  - Transactional
  - Trust
Capability Specification

- Non functional properties
- Imported Ontologies
- Used mediators
  - *OO Mediator:* importing ontologies as terminology definition
  - *WG Mediator:* link to a Goal that is solved by the Web Service
- Pre-conditions
  - What a web service expects (conditions over the input)
- Assumptions
  - Conditions on the state of the world before the WS execution
- Post-conditions
  - The result of the WS in relation to the input, and conditions on it
- Effects
  - Conditions on the state of the world after the WS execution
    (i.e. changes in the state of the world)
Choreography in WSMO

“Choreography describes the behavior of the service from a user point of view”

- **External Visible Behavior**
  - those aspects of the workflow of a Web Service where User Interaction is required
  - described by process / workflow constructs

- **Communication Structure**
  - messages sent and received
  - their order (messages are related to activities)
Choreography in WSMO (2)

- **Grounding**
  - Concrete communication technology for interaction
  - Choreography related errors (e.g. input wrong, message timeout, etc.)
- **Formal Model**
  - Allow operations / mediation on Choreographies
  - Formal Basis: Abstract State Machines (ASM)
WSMO Orchestration

“...how the overall functionality of the service is achieved by the cooperation of other WSMO service providers”

- **Orchestration Language**
  - Decomposition of Web Service functionality
  - Control structure for aggregation of Web Services

- **Web Service Composition**
  - Combine Web Services into higher-level functionality
  - Resolve mismatches occurring between composed Web Services

- **Proxy Technology**
  - Placeholders for used Web Services
  - Facility for applying the Choreography of used Web Services
WSMO Orchestration Overview

Decomposition of the Web Service functionality into sub-functionalities

Proxies as placeholders for used Web Services

Control Structure for aggregation of other Web Services
Choreography & Orchestration Example

• VTA example:

• WSMO Choreography models all visible interactions of the service (Orchestration shows how all the interaction are related)
WSMO Mediators

Objectives that a client may have when consulting a Web Service

Provide the 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:** mediate heterogeneous **Data Sources**
  2. **Process/Protocol Level:** mediate heterogeneous Business Processes/Communication Patterns
WSMO Mediators Overview
Mediator Structure

WSMO Mediator
uses a Mediation Service via
Target Component
- as a Goal
- directly
- through another mediator

Mediation Services

Source Component

Source Component

1..n
GG Mediators

- **Aim:**
  - Support specification of Goals by re-using existing Goals
  - Allow definition of **Goal Ontologies** (collection of pre-defined Goals)
  - Terminology mismatches handled by OO Mediators

- **Example: Goal Refinement**
  
  ![Diagram of GG Mediator]

  - **Source Goal**
    - “Buy a ticket”
  - **GG Mediator**
    - Mediation Service
  - **Target Goal**
    - “Buy a Train Ticket”
**WG & WW Mediators**

- **WG Mediators:**
  - link a Web Service to a Goal and resolve occurring mismatches
  - match Web Services and Goals that do not match a priori
  - handle terminology mismatches between Web Services and Goals
  ⇒ broader range of Goals solvable by a Web Service

- **WW Mediators:**
  - enable interoperability of heterogeneous Web Services
  - handle terminology mismatches between Web Services
  ⇒ support automated collaboration between Web Services
  - Data Mediation for resolving terminology mismatches (OO Mediators)
  - Process/Protocol Mediation for establishing valid multi-party collaborations and making Business Processes interoperable
Web Services Modelling Language (WSML)

Adrian Mocan
WSML - Web Service Modeling Language

- WSML provides a formal grounding for the conceptual elements of WSMO, based on:
  - Description Logics
  - Rule Languages
  - First-Order Logic
Rationale of WSML

- Provide a Web Service Modeling Language based on the WSMO conceptual model
  - Concrete syntax
  - Semantics
- Provide a Rule Language for the Semantic Web
- Many current Semantic Web languages have
  - undesirable computational properties
  - unintuitive conceptual modeling features
  - inappropriate language layering
    - RDFS/OWL
    - OWL Lite/DL/Full
    - OWL/SWRL
Variants of WSML

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

WSML-Core \(\rightarrow\) WSML-Flight \(\rightarrow\) WSML-Rule

Description Logics

Logic Programming

First-Order Logic (with nonmonotonic extensions)
WSML Conceptual Syntax for Ontologies

- Ontologies
- Namespaces
- Imported Ontologies
- Used Mediators

- Concepts
- Relations
- Functions
  - Special kind of relation
- Instances
  - Explicitly defined in ontology
  - Retrieved from external instance store
- Axioms

Extra-Logical declarations

Logical Declarations

Non-Functional Properties
WSML Logical Expressions

- Frame- and first-order-based concrete syntax (BNF Grammar in D2, Appendix B)
- Elements:
  - Function symbols (e.g. $f()$)
  - Molecules (e.g. Human subClassOf Animal, John memberOf Human, John[name hasValue ‘John Smith’]).
  - Predicates (e.g. distance(to:?x, from:?y, distance:?z))
  - Logical connectives (or, and, not, implies, equivalent, impliedBy, forall, exists)
- Example:
  
  ```
  ?x memberOf Human equivalent
  ?x memberOf Animal and ?x memberOf LegalAgent.
  ```
WSML Goals and Web Services

• Goal / Web Service
  – assumptions
  – effects
  – pre-conditions
  – post-conditions

are defined through WSML logical expressions

• Logical expressions are based on ontologies
**WSML-Flight - Example**

### Conceptual Syntax

```plaintext
concept ticket
  origin ofType location
  destination ofType location
  departure ofType xsd:dateTime
  arrival ofType xsd:dateTime
  fare ofType price
```

### Logical Expression Syntax

```plaintext
axiom validDates
  definedBy
  <- ?xmemberOf ticket[arrival hasValue ?y, departures hasValue ?z] and ?y < ?z.
```
WSML Summary

• Formal languages for WSML
• Variants:
  – WSML-Core
  – WSML-Flight
  – WSML-Rule
  – WSML-DL
  – WSML-Full
• Modular, Frame-based
• Conceptual syntax vs. Logical Expressions
• Syntaxes:
  – Human readable
  – XML
  – OWL/RDF
Using WSMO to address Web Services problems

Adrian Mocan
• “Web service” and “service” have to be distinguished:
  – *Web service*: a computational entity able to perform many services, e.g. Amazon Web service
  – *Service*: a concrete invocation of a Web service, e.g. buying „Silver Bullet“ for EUR 37,40 with free delivery within 2-3 days.

• Heuristic Classifications (William J. Clancey, 1985)
  – *Abstraction*
    • Process of translating concrete descriptions into features usable for classification, e.g. a concrete body temperature into „low fever“
  – *Matching*
    • Inferring potential classification or solutions from extracted features
  – *Refinement*
    • Inferring final diagnoses; it may include the acquisition of new features describing the given case
WSMO Discovery

Abstracted Findings → Matching → Abstracted Diagnosis

Abstraction

Findings

Refinement

Diagnosis

Data
Process
Abstracting goals from concrete user desire, e.g.: „Buying a train ticket from Innsbruck to Karlsruhe for today” into “buying train tickets in Europe”.
Abstracting goals from concrete user desire, e.g.: “Buying a train ticket from Innsbruck to Karlsruhe for today” into “buying train tickets in Europe”.

Matching between abstract goals and abstract services, e.g. “train tickets in Europe” and “transportation in Europe”.
WSMO Discovery

Based on the use of a Web service to discover the actual service. Requires strong mediation (protocol, process and data).

Matching between abstract goals and abstract services, e.g. “train tickets in Europe” and “transportation in Europe”

Abstracting goals from concrete user desire, e.g.: “Buying a train ticket from Innsbruck to Karlsruhe for today” into “buying train tickets in Europe”.
WSMO Discovery

Abstracting goals from concrete user desire, e.g.: “Buying a train ticket from Innsbruck to Karlsruhe for today” into “buying train tickets in Europe”.

Matching between abstract goals and abstract services, e.g. “train tickets in Europe” and “transportation in Europe”.

Based on the use of an Web service to discover the actual service.
Requires strong mediation (protocol, process and data)
Description and Discovery

Capability descriptions - Levels of abstraction & possible accuracy

What? (Syntactically)

→ Syntactic capability
  perhaps complete & perhaps correct

What? (Semantic „Light“)

→ Abstract capability
  complete & perhaps correct

What & When? (Semantic „Heavy“)

→ Concrete capability
  complete & correct
  (if user input known & interaction)

{Keyword}

WS

Level of Abstraction
“Automated selection, composition, and interoperation of [existing] Web services to perform some complex task, given a high-level description of an objective.”

- Web services are described at two abstraction levels:
  - functional (or capability) level
    - the focus is on the service inputs, outputs, preconditions, and effects
    - WSMO capability model
  - process level
    - the Web service is defined by an activity flow or an interaction pattern
    - WSMO interface model
Functional-level vs. process-level
- Composition task -

• Functional-level composition
  – select a set of services that, combined in a suitable way, are able to match a given objective:
    • Given the requirements for a trip (destination, duration, budget…), find the services that are necessary to prepare the trip (Deutsche Bahnhof, Hotels@Karlsruhe, Hertz…)

• Process-level composition
  – define an interaction pattern with the selected services, so that an executable implementation of the composition is obtained:
    • Find the correct order for the interactions with the selected services (e.g., interactions with train and hotel have to be interleaved to guarantee consistency of arrival and departure dates)
Service Grounding – WSMO

• Deal with existing WSDL services
  – Map from XML Schema used in WSDL to WSMO
  – Use existing tools to mediate from WSMO ontology to WSMO ontology

• Also investigating
  – Using XSLT to map from XML-S of WSDL directly to WSML/XML of ontology used by WSMO description

• Ultimate aim to have **semantic** description of interface grounding in the choreography
Service Grounding – WSMO

1. Create WSMO description

2. Map XML schema to WSMO conceptual model

3. Create Mapping Rules

4. Use mapping rules from WSMO choreography

Amazon WS

- WSDL
- XML Schema

WSMO WS

Interface

Book Ontology

Mapping Rules

WSMO ontology from XML Schema

used by
Conclusion: How WSMO Addresses WS problems

- **Discovery**
  - Provide formal representation of capabilities and goal
  - Conceptual model for service discovery
  - Different levels to Web Service discovery
- **Composition**
  - Provide formal representation of capabilities and choreographies
- **Invocation**
  - Support any type of WS invocation mechanism
  - Clear separation between WS description and implementation
- **Guaranteeing Security and Policies**
  - No explicit policy and security specification yet
  - Proposed solution will interoperate with WS standards
- **Mediation and Interoperation**
  - Mediators as a key conceptual element
  - Mediation mechanism not dictated
  - (Multiple) formal choreographies + mediation enabled interoperation
- The solutions are envisioned maintaining a strong relation with existing WS standards
Web Service Execution Environment (WSMX)

Michal Zaremba
Overview

- WSMX Overview
- Components and System Architecture
- Interrelationship of components
  - Execution semantics
- Component interfaces
  - Data flow between components
WSMX Introduction

- WSMX is a software framework that allows runtime binding of service requesters and service providers
- WSMX interprets service requester goal to
  - Discover matching services
  - Select the service that best fits
  - Provide data mediation if required
  - Make the service invocation
- WSMX is based on the conceptual model provided by WSMO
- WSMX has a formal execution semantics
- WSMX has service oriented and event-based architecture based on microkernel design using such enterprise technologies as J2EE, Hibernate, Spring, JMX, etc.
WSMX 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
Scope of WSMX Development

- Reference implementation for WSMO
- Complete architecture for SWS discovery, mediation, selection and invocation
- Example of implemented functionality - achieving a user-specified goal by invoking WS described with the semantic markup
System Architecture

2005 OASIS SYMPOSIUM
Dynamic Execution Semantics

- WSMX consists of loosely coupled components
- Components might be dynamically plug-in or plug-out
- Execution Semantics - invocation order of components
- Event-based implementation
- New execution semantics can appear in the future including new components
- We need a flexible way to create new execution semantics and deploy them in the system
- Ultimate goal is to execute workflow definition describing interactions between system components
Define “Business” Process

- Discover Web Services
- Discover Services
- Mediate Data
- Return Mediated Data
- Return Web Services
- Call Invoker
- Confirmed
- Check Choreography
- Confirmed
- Data Mediator Wrapper
- Created
- Choreography Wrapper
- End
- Registry of known components
Event-based Implementation

“Business” Process – Internal Workflow

Event and Notification Distribution/Delivery Mechanism

Core – Manager

Choreography Wrapper

Discovery Wrapper implements Mediator Interface

Data Mediator Wrapper

Communication Manager Wrapper

Choreography

Discovery

Mediator

Communication Manager
System Architecture

WSMX

WSMXML Manager

WSMXML Manager Core

CM Wrapper
PM Wrapper
Parser Wrapper
Discovery Wrapper
Selector Wrapper
OM Wrapper
PM Wrapper
Choreography Wrapper
Orchestration Wrapper

Resource Manager Interface
WSMO Objects Datastore
NonWSMO Objects Datastore

Reasoner Interface
Reasoner
Reasoner
Reasoner

Datastore
Datastore
Flora/KSB

Service Requesters
Back-end application 1
Back-end application 2
...
Back-end application n
Agent 1 acting on behalf of user a
Agent 2 acting on behalf of user b
Agent 3 acting on behalf of user c

Adapters
Adapter 1
Adapter 2
...
Adapter n

Data and Communication Protocols

Service Providers
Web Service 1
Web Service 2
...
Web Service n
Request to discover Web services. May be sent to adapter or adapter may extract from backend app.
System Architecture

Goal expressed in WSML sent to WSMX System Interface
System Architecture

Comm Manager component implements the interface to receive WSML goals
System Architecture
System Architecture

Choreography wrapper picks up event for Choreography component
System Architecture

A new choreography instance is created.
System Architecture

Core is notified that choreography instance has been created.
System Architecture

Parser wrapper picks up event for Parser component
System Architecture

WSML goal is parsed to internal format
System Architecture
System Architecture
System Architecture

Discovery is invoked for parsed goal
System Architecture

Discovery component requires data mediation.
System Architecture
System Architecture
After data mediation, discovery component completes its task.
System Architecture

2005 OASIS SYMPOSIUM

WSMX

WSMX Manager

WSMX Manager Core

CM Wrapper
RM Wrapper
OM Wrapper
Parser Wrapper
Selector Wrapper
Discovery Wrapper
Data Manager Interface
Selector Interface
Data Manager Interface
Selector Interface
Data Manager Interface
Selector Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestration Interface
Orchestr
After discovery, the choreography instance for goal requester is checked for next step in interaction.
System Architecture
System Architecture
System Architecture

Next step in choreography is to return set of discovered Web services to goal requester.
System Architecture

Set of Web Service descriptions expressed in WSML sent to appropriate adapter
System Architecture

Set of Web Service descriptions expressed in requester’s own format returned to goal requester
WSMX Summary

- Event based component architecture
- Conceptual model is WSMO
- End to end functionality for executing SWS
- Has a formal execution semantics
- Open source code base at sourceforge
- Developers welcome
WSMX Useful Links

- Home
  - http://www.wsmx.org/
- Overview
  - http://www.wsmo.org/2004/d13/d13.0/v0.1/
- Architecture
  - http://www.wsmo.org/2004/d13/d13.4/v0.2/
- Mediation
  - http://www.wsmo.org/2004/d13/d13.3/v0.2/
- Execution Semantics
  - http://www.wsmo.org/2004/d13/d13.2/v0.1/
- Open source code base at SourceForge
  - https://sourceforge.net/projects/wsmx
IRS-III: A framework and platform for Semantic Web Services

Liliana Cabral
The Internet Reasoning Service is an infrastructure for publishing, locating, executing and composing Semantic Web Services, organized according to the WSMO conceptual model.
IRS-III Framework

IRS-III Server

- Domain Models
- Goal Descriptions
- Web Service Descriptions + Registry of Implementors
- Mediator Descriptions

IRS Publisher
- Lisp
- Java
- Java WS

IRS Client

SOAP
IRS-III Features

- Provides *capability-centred* service invocation
- Provides built-in brokering and service discovery support
- Publishing support for variety of platforms
  - Java Web Services, Java, Lisp, Web Applications
- Enables publication of ‘standard code’
  - Provides clever wrappers automatically, which turn code into web services
  - One-click publishing of web services
- Provides Java API for client applications
- Based on Soap messaging standard
IRS-III Architecture

Publishing Platforms

Web Service
Java Code
Web Application

WSMX
Browser
Publishing Clients
Invocation Client

LispWeb Server

SOAP
Browser Handler
Publisher Handler
Invocation Handler

WS Publishers Registry
OCML
WSMO Library

OWL(-S) Handler

SOAP Handler

Publishing Platforms

WSMXML
Publishing Platform Architecture

Publishing Clients

SOAP

IRS-III Server

Invocation Client

HTTP Server

IRS-III Publishing Platform

SOAP Handler

Service Registrar

WS Service Registry

Service Invoker

Web Service 1

Web Service 2

Web Service 3
IRS-III/WSMO differences

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

Development Team

English Train Ticket Booking Web Service
  B Austrian Train Ticket Booking Web Service
  B German Train Ticket Booking Web Service
  B French Train Ticket Booking Web Service

Customer Team

Customer

Buy book goal
Book Flight goal
Buy cinema tickets goal
Buy train ticket goal
SWS Creation & Usage Steps

- Create a goal description
  - (e.g. book-train-goal)
  - Add input and output roles
  - Include role type and soap binding
- Create a wg-mediator description
  - Link a goal to a Web Service
  - Source component = goal
  - Possibly add a mediation service
- Create a web service description
  - Used-mediator of WS capability = wg-mediator above
- Publish Lisp function against web service description
- Invoke web service by ‘achieve goal’
Multiple Web Services for a Goal

• Each WS links to a Goal through the mediator in the used-mediator slot of capability
  – Some WS may share a mediator

• Define a constraint for solving the Goal - a logical expression for assumption slot of WS capability
  – logical expression format
    • (kappa (?goal) <ocml relations>)
  – Getting the value of an input role
    • (wsmo-role-value ?goal <role-name>)}
Valid Logical Expressions (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
- Example:
  (kappa (?goal)
   (member (wsmo-role-value ?goal 'has_source_currency) '(euro pound)))

Defining a WG-Mediator

passenger (person)
departure (city)
destination (city)
time-date (list)

Source

WGMediator

Mediation Service

time-date (list)
time-date (list)

G

G

time-date (univ-time)
time-date (univ-time)

WS

Target
Defining a Mediation Service

- Defined in the Mediator
- Mediation-service = Goal
- Web Service implements the mediation (mappings)
- Mediation Goal input roles are a subset of source Goal input roles
- Mediation Goal output is a subset of target Web Service input roles.
Goal Based Invocation

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

**Invocation**
- Invoke selected web service
  - European-exchange-rate
IRS-III Demo

Liliana Cabral
European Travel Scenario
European Travel Demo
Demo - Objective

- Develop an application for the European Travel scenario based on SWS. The application should support a person booking a train ticket between 2 European cities at a specific time and date.
- Create Goal, Web service and Mediator WSMO descriptions in IRS-III (european-travel-service-descriptions) for available services. Service constraints involves start and end locations and the type of traveller. Use the assumption slot to express this.
- Publish available lisp functions against Web Service descriptions.
- Invoke the web services through ‘Achieve Goal’
- Solution using IRS-III browser will be provided.
Travel Related Knowledge Models
Is-in-country <city> <country> e.g. (is-in-country berlin germany) -> true
student instances: john matt michal
business-person instances: liliana michael
Goals

• 1- Get train timetable
  – Inputs: origin and destination cities, date
  – Output: timetable (list)

• 2- Book train
  – Inputs: passenger name, origin and destination cities, departure time-date
  – 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 476 going from MILTON-KEYNES to LONDON at 16:34, 15, SEPTEMBER 2004.
The price is 179 Euros."

3- book-french-train-journey

*sinuhe paris lyon (3 4 6 18 8 2004)*

"SNCF: SINUHE is booked on the 593 going from PARIS to LYON at 6:12, 18, AUGUST 2004.
The price is 25 Euros."
Available Functions (2/3)

4- book-german-train-journey
cristoph berlin frankfurt 3305020023
"German Rail (Die Bahn): CHRISTOPH is booked on the 362 going from BERLIN to FRANKFURT at 14:47, 24, SEPTEMBER 2004. The price is 35 Euros."

5- book-austrian-train-journey
sinuhe vienna innsbruck 3304686609
"Austrian Rail (OBB): SINUHE is booked on the 681 going from VIENNA to INNSBRUCK at 17:43, 20, 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 408 going from LONDON to
The price is 86 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
Using IRS-III Browser for the VTA Demo application

• Semantic Descriptions of:
  – Goals
  – Web Services
  – Mediators

• Publishing

• Invocation
IRS-III Browser

**Goal Book-Train-Goal (European-Travel-Service-Descriptions)**

- **Parent Goal:** Goal
- **Associated Web Services:** book-french-train-ws
  (european-travel-service-descriptions), book-german-train-ws
  (european-travel-service-descriptions)
- **Input Roles:**
  - Passenger
    - **Type:** Person
  - Has-Departure
    - **Type:** City
Creating a Goal description
Creating a Mediator description

[Image of a screenshot showing a mediator editor with details for a subtitle mediator:]
- **Name**: book-german-train-mediator
- **Ontology**: european-travel-service-descriptions
- **Mediator Parent**: wg-mediator
- **Source Component**: book-train-goal
- **Target Component**: 
- **Mediation Service**: book-german-train-mediation-service-goal
- **Reduction**: 

[Save, Delete, Cancel buttons]
Creating a Web Service description

![Web Service Editor](image)

- **Name**: book-german-train-ws
- **Ontology**: european-travel-service-descriptions
- **Parent**: WEB-service

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>has-time-date</td>
<td>universal-time</td>
</tr>
</tbody>
</table>

### Output

- **Name**: 
- **Type**: 

**Controls**: Save, Delete, Cancel
Adding a Mediator to the Web Service Capability
Adding a constraint to the Web Service Capability
Creating a Goal (Mediation Service)
Creating a Mediator description (Mediation Service)
Adding a Mediator to the Web Service (Mediation Service)
Publishing Web Services
(lisp functions)
Achieving a Goal (Mediation Service)
Achieving a Goal

Goal Name: book-train-goal
Goal Ontology: european-travel-service-descriptions

Input-Roles:
- Name: passenger, Value: christoph
- Name: has-departure, Value: berlin
- Name: has-destination, Value: frankfurt
- Name: has-time-date, Value: (01 09 10 12 2004)

Response:
German Rail (Die Bahn): CHRISTOPH is booked on the 453 going from BERLIN to FRANKFURT at 9:59, 10, DECEMBER 2004. The price is 32 Euros.
IRS-III Future Work

• IRS-III Choreography definition language is being specified.
  – Based on guarded state transitions as forward chaining rules
• IRS-III Orchestration is being defined.
• OO-mediators will have mapping rules.
IRS-III Link

- Download available:
  - Java API
  - Browser/Editor
WSMO Tools
(in development)

   Java API for WSMO / WSML
4. **WSMT – Web Services Modelling Toolkit**
   (currently: SWWS Studio)
   Creation and editing of WSMO specifications
   WSML Editor
   Ontology Management System OMS
   Open for Plug-Ins for SWS tools (discovery, composer, …)
6. **WSML Validator and Parser**
   validates WSMO specifications in WSML
   parsing into intermediary FOL format (every FOL compliant syntax can be derived from this)
7. **OWL Lite Reasoner for WSML-OWL variant**
   OWL Lite Reasoner based on TRIPLE
Summary, Conclusions & Future Work

Liliana Cabral
Conclusions

• This tutorial should enable you to:
  – understand aims & challenges within Semantic Web Services
  – understand the objectives and features of WSMO
  – model Semantic Web Services with WSMO
  – correctly assess emerging technologies & products for Semantic Web Services
  – use implemented tools to create SWS
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)
- In regard of WSMO languages: WSML Working Group: [http://www.wsml.org](http://www.wsml.org)
- WSMO implementation: WSMX working group can be found at: [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 WSMO


- [WSMO Use Case]: Stollberg, M.; Lara, R. (ed.): *WSMO Use Case Modeling and Testing*, WSMO Working Drafts D3.2; D3.3.; D3.4; D3.5, final version 0.1, 17 November 2004.
References WSMO

References WSMX

- Matthew Moran and Michal Zaremba and Adrian Mocan and Christoph Bussler: *Using WSMX to bind Requester & Provider at Runtime when Executing Semantic Web Services*, 1st WSMO Implementation Workshop, Sep, 2004, Frankfurt, Germany.
References IRS-III


Acknowledgements

The WSMO work is funded by the European Commission under the projects DIP, Knowledge Web, SEKT, SWWS, AKT and Esperonto; by Science Foundation Ireland under the DERI-Lion project; and by the Vienna city government under the CoOperate program.