Semantic Web Services: The Web Service Modelling Ontology and IRS-III

John Domingue

Knowledge Media Institute, The Open University, UK
Contents

• Semantic Web Services: Problem and Vision
• Web Services Modelling Ontology
• Comparison with OWL-S (short)
• Overview of IRS-III
• Summary
What’s a Web Service?

• A program programmatically accessible over standard internet protocols
• Loosely coupled, reusable components
• Encapsulate discrete functionality
• Distributed
• Add new level of functionality on top of the current web
Web Services Framework

- **UDDI Registry**
  - Points to Service
  - Finds Service

- **Service Consumer**
  - Communicates with XML Messages

- **WSDL**
  - Describes Service
  - Points to Description

- **Web Service**
  - SOAP
What’s the big deal?

• In *U.S. Web Services Market Analysis, 2002* IDC predicts that Web services will become the dominant distributed computing architecture in the next 10 years. Web services will drive software, services and hardware sales of $21 billion in the U.S. by 2007 and will reach $27 billion in 2010.

• Web services promise easy access to remote content and application functionality, independently of the provider's platform, the location, the service implementation, or the data format. *Kuassi Mensah, Oracle*

• Exposure of capabilities
Sample Applications

The following is a list of some of the best applications that use Amazon Web Services. Use them as inspiration for your own work with the platform.

Tools for Associates

- **Amazon ad HTML** - Associate Engine allows Associates to quickly and easily add thousands of Amazon.com products to their website using Amazon.com's XML Web Services. Associate Engine is not difficult to use. The good news is that you do not have to know any complicated programming languages (such as AWS, perl, PHP, SOAP, XML, XSLT). All that is required is that you know how to create a simple HTML webpage and how to insert links in your webpage.

- **Piranha Feed** - Piranha Feed provides flexible, fast, and easy product link integration PHP scripts that can be embedded into your existing website. Some of the advantages of using Piranha Feed include: Amazon product listings on your site are automatically updated, can be plugged into existing PHP, HTML, XML, CGI, and ASP pages with ease, and uses a sophisticated caching algorithm to further save bandwidth and maximize speed.

- **AssociatedShop.com** - Are you running a website and always thought about offering products related to the topic of your website without any risk for a nice referral fee? With our free, hosted shop Associates can offer an expansive selection of high-quality products from Amazon.com and Amazon.co.uk, while earning up to 15% referral fees on every sold product. Just select what you want to sell, for example tell us a keyword like "britney spears" - all the rest is done by our shop-system and Amazon. Configure your shop and then just modify your existing webpages to link to it. It's that easy!

- **Amazon Products Feed** - Free open source perl script that utilizes Amazon.com's Web Services to provide a real-time listing of products on your site. The script provides links for your visitors to purchase these items on Amazon.com using your affiliate code or associate id. This script parses Amazon.com's XML feed to provide search results and bestseller listings in HTML for your visitors. It supports both Amazon.com and Amazon.co.uk.

- **MyBookVendor** - Easy to use scripts: a scalable, fully customizable book listing program to allow Associates to create a bookstore out of pre-generated book lists, a highly customizable search box, allowing Amazon.com affiliates to create custom bookstore pages, returning Amazon.com search results directly to their web site, while maintaining a consistent look and feel, and a small, stand-alone server side include to promote featured Amazon.com titles on an Associate's website.

- **MailSocket** - Associates can install MailSocket into their site with a single line of HTML code. Mail Socket Features: hundreds of thousands of products from 13 separate stores, browse bar to help your customers keep track of their path through the mall as they drill down to find the products that they need, and your customers can add items to a shopping cart and then check out at Amazon.com using your Amazon Associates ID, so you earn the commissions.
Problems with Web Services Today

• Descriptions are syntactic
• All tasks associated with web services application development have to be carried out by humans:
  – discovery, composition and invocation
• Problems of scalability
Larry Says

Semantic differences remain the primary roadblock to smooth application integration, one which Web Services alone won't overcome. 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 problem is not in the plumbing - it’s in the semantics”

Mike Brodie, Chief Scientist
Verizon
The Third Summer School on Ontological Engineering and the Semantic Web (SSSW'05)

SWS Vision

Dynamic

Static

Web Services
(UDDI, WSDL, SOAP)

Semantic Web Services

Web
(URI, HTML, HTTP)

Semantic Web
(RDF, OWL)

Syntax

Semantics
Semantic Web Services (is)

• Semantic Web Technology
  – Machine readable data
  – Ontological basis

Applied to

• Web Services Technology
  – Reusable computational resources

To automate all aspects of application development through reuse
SWS Activities (1/2)

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
SWS Activities (2/2)

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
Web Service Modelling Ontology (WSMO)
WSMO is ..

• a conceptual model for Semantic Web Services:
  – Ontology of core elements for Semantic Web Services
  – a formal description language (WSML)
  – execution environment (WSMX and IRS-III)

• … derived from and based on the Web Service Modeling Framework WSMF

• a SDK-Cluster Working Group
  (joint European research and development initiative)
SDK-Cluster

- **SEKT** (Semantically-Enabled Knowledge Technologies)
  http://sekt.semanticweb.org/

- **DIP** (Data, Information and Process with Semantic Web Services)
  http://www.nextwebgeneration.org/projects/dip/

- **Knowledge Web**
  http://knowledgeweb.semanticweb.org/

- **SDK – Cluster**
  http://www.sdk-cluster.org/
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

- Web Compliance
- Ontology-Based
- Strict Decoupling
- Centrality of Mediation
- Ontological Role Separation
- Description versus Implementation
- Execution Semantics
WSMO Top Level Notions

Objectives that a client wants to achieve by using Web Services

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
Non-Functional Properties

every WSMO elements is described by properties that contain relevant, non-functional aspects

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

Objectives that a client wants to achieve by using Web Services

- 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 Usage & Principles

• Ontologies are used as 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:

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

Objectives that a client wants to achieve by using Web Services

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)
Capability Specification

- Non functional properties
- Imported Ontologies
- Used mediators
  - OO Mediator: importing ontologies with mismatch resolution
  - WG Mediator: link to a Goal wherefore service is not usable a priori
- Pre-conditions
  What a web service expects in order to be able to provide its service. They define conditions over the input.
- Assumptions
  Conditions on the state of the world that has to hold before the Web Service can be executed
- Post-conditions
  describes the result of the Web Service in relation to the input, and conditions on it
- Effects
  Conditions on the state of the world that hold after execution of the Web Service (i.e. changes in the state of the world)
WSMO Web Service Description

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

Capability
- Advertising of Web Service
- Support for WS Discovery

WS - Advertising of Web Service
- Support for WS Discovery

Web Service Implementation
- complete item description
- quality aspects
- Web Service Management
- Advertising of Web Service
- Support for WS Discovery

Choreography --- Service Interfaces --- Orchestration

DC + QoS + Version + financial

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

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

The Third Summer School on Ontological Engineering and the Semantic Web (SSSW'05)
Choreography and 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 Aspects

*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**
  - concrete communication technology for interaction
  - choreography related errors (e.g. input wrong, message timeout, etc.)

- **Formal Model**
  - reasoning on Web Service interfaces (service interoperability)
  - allow mediation support on Web Service interfaces
Orchestration Aspects

Control Structure for aggregation of other Web Services

- decomposition of service functionality
- all service interaction via choreographies
Service Interface Description

- Ontologies as data model:
  - all data elements interchanged are ontology instances
  - service interface = evolving ontology

- Abstract State Machines (ASM) as formal framework:
  - dynamics representation: high expressiveness & low ontological commitment
  - core principles: state-based, state definition by formal algebra, guarded transitions for state changes
  - overcome the “Frame Problem”

- further characteristics:
  - not restricted to any specific communication technology
  - ontology reasoning for service interoperability determination
  - basis for declarative mediation techniques on service interfaces
Service Interface Description Model

• 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 (action)
  – different for Choreography and Orchestration
Service Interface Example

Communication Behavior of a Web Service

State $\omega_1$

$$a \text{memberOf} A \left[ \begin{array}{l} \text{att1 hasValue} x \\ \text{att2 hasValue} y \end{array} \right]$$

received ontology instance $a$

Guarded Transition $GT(\omega_1)$

$$\begin{array}{l}
\text{IF} (a \text{memberOf} A \left[ \begin{array}{l} \text{att1 hasValue} x \end{array} \right], \\
\text{THEN} \\
(b \text{memberOf} B \left[ \begin{array}{l} \text{att2 hasValue} m \end{array} \right])
\end{array}$$

State $\omega_2$

$$\begin{array}{l}
a \text{memberOf} A \left[ \begin{array}{l} \text{att1 hasValue} x, \\
\text{att2 hasValue} m \end{array} \right] \\
b \text{memberOf} B \left[ \begin{array}{l} \text{att2 hasValue} m \end{array} \right]
\end{array}$$

sent ontology instance $b$

Vocabulary:
- Concept A in $\Omega_{in}$
- Concept B in $\Omega_{out}$
WSMO Goals

Objectives that a client wants to achieve by using Web Services

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

- Ontological De-coupling of Requester and Provider
- Derived from task / problem solving methods/domain model
- Structure and reuse of requests
  - Search
  - Diagnose
  - Classify
  - Personalise
  - Book a holiday
- Requests may in principle not be satisfiable
- Ontological relationships & mediators used to link goals to web services
Goal Specification

- Non functional properties
- Imported Ontologies
- Used mediators
  - **OO Mediators**: importing ontologies with heterogeneity resolution
  - **GG Mediator**:
    - Goal definition by reusing an already existing goal
    - allows definition of **Goal Ontologies**
- Requested Capability
  - describes service functionality expected to resolve the objective
  - defined as capability description from the requester perspective
- Requested Interface
  - describes communication behaviour supported by the requester for consuming a Web Service (Choreography)
  - Restrictions / preferences on orchestrations of acceptable Web Services
WSMO Mediators

Objectives that a client wants to achieve by using Web Services

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 …
  – For 1$ on programming, $5 - $9 on integration
  – Mismatches on structural / semantic / conceptual / level
  – Assume (nearly) always necessary

• Description of role
  – Components that resolve mismatches
  – Declarative description of arbitrary web service

• Types of Mediation within Semantic Web Services:
  (1) Data: mediate heterogeneous Data Sources
  (2) Protocol: mediate heterogeneous Communication Patterns
  (3) Process: mediate heterogeneous Business Processes
WSMO Mediators Overview
Mediator Structure

WSMO Mediator
uses a Mediation Service via
- as a Goal
- directly
- optionally incl. Mediation

Source Component

Source Component

Target Component

1 .. n

Mediation Services
OO Mediator - Example

Merging 2 ontologies

Train Connection Ontology (s1) → OO Mediator Mediation Service → Purchase Ontology (s2)

Goal: "merge s1, s2 and s1.ticket subclassof s2.product"

Discovery → Mediation Services → Train Ticket Purchase Ontology
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**

Source Goal
“Buy a ticket”

GG Mediator Mediation Service

Target Goal
“Buy a Train Ticket”

postcondition:
“aTicket memberof trainticket”
WG & WW Mediators

• **WG Mediators:**
  – link a Web Service to a Goal and resolve occurring mismatches
  – match Web Service 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
  ⇒ support automated collaboration between Web Services

  – **OO Mediators** for terminology import with data level mediation
  – Protocol Mediation for establishing valid multi-party collaborations
  – Process Mediation for making Business Processes interoperable
OWL-S
OWL-S Ontology

- OWL-S is an OWL ontology to describe Web services
- OWL-S leverages on OWL to
  - Support capability based discovery of Web services
  - Support automatic composition of Web Services
  - Support automatic invocation of Web services
  - OWL-S provides a semantic layer over Web services standards
    - OWL-S relies on WSDL for Web service invocation (see Grounding)
    - OWL-s Expands UDDI for Web service discovery (OWL-S/UDDI mapping)
OWL-S Upper Ontology

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

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

- Capability specification
- General features of the Service
  - Quality of Service
  - Classification in Service taxonomies
WSMO OWL-S Comparison

- Historical
  - OWL-S planning (agents)
  - WSMO knowledge modelling and B2B integration
- Representation
  - OWL-S based on OWL
  - WSMO on WSML family
- WSMO explicit conceptualisation of user context
- WSMO explicit conceptualisation of mediation
- WSMO Interfaces ≈ process model
  - WSMO provides choreography + orchestration while OWL-S provides only orchestration
  - WSMO service interface description model with ASM-based formal semantics
  - OWL-S formal semantics has been developed in very different frameworks such as Situation Calculus, Petri Nets, Pi-calculus
  - OWL-S Process Model is extended by SWRL / FLOWS
- OWL-S Grounding ≈ current WSMO Grounding
IRS-III: A framework and platform for building Semantic Web Services
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

SOAP
IRS-III Architecture

LispWeb Server

Publishing Platforms

SOAP

Browser Handler
Publisher Handler
Invocation Handler

WS Publisher Registry
OCML
WSMO Library

OWL(-S) Handler

Web Service
Java Code
Web Application

JAXA

Publishing Platforms

SOAP

Browser
Publishing Clients
Invocation Client

WSMX

The Third Summer School on Ontological Engineering and the Semantic Web (SSSW'05)
Publishing Platform Architecture

IRS-III Publishing Platform

HTTP Server

WS Service Registry

Service Registrar

Service Invoker

Web Service 1
Web Service 2
Web Service 3

Publishing Clients

Invocation Client

IRS-III Server

SOAP Handler

SOAP

SOAP

SOAP
IRS-III/WSMO differences

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

• Web Services are
  – Reusable programs available over the web
  – Match business services

• Semantic web services
  – Applies semantic web technology to web services

• WSMO
  – Ontology, Goal, Web Service and Mediator
  – Ontological separation of requester and provider context
  – Mediation as first class citizen

• IRS-III
  – One click publishing
  – Capability based invocation
  – Implements WSMO
IRS-III References


WSMO References

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

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

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


WSMO References


WSMO References

WSMO References

References OWL-S

• The main repository of papers on OWL-S is at http://www.daml.org/services/owl-s/pub-archive.html

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

IRS-III was developed within the AKT, MIAKT and DIP projects. The IRS-III team also includes: Liliana Cabral, Stefania Galizia, Vlad Tanasescu, Alessio Gugliotta and Enrico Motta

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.
Semantic Web Services
Hands-On Session
with IRS-III

John Domingue and Liliana Cabral

Knowledge Media Institute,
The Open University, UK
European Travel Scenario
European Travel Demo
IRS-III Hands On Task

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

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

- Publish available lisp functions against your descriptions.

- Invoke the web services.

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

IRS Server (3000)

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

Travel Services (3001)

IRS Lisp Publisher

IRS-III Knowledge Model Browser & Editor
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
Example: Multiply Goal
Example: Multiply Mediator
Example: Multiply Web Service (1/2)
Example: Multiply Web Service (2/2)
Example: Publishing for Multiply
Example: Invocation Multiply Goal
IRS-III Visualizer

Service: IRS Server
Received Achieve Goal Message
Ontology: wsmo-test
Goal Type: multiply-goal
has-operand1 5
has-operand2 6

Calling service Multiply-Web-Service (Mu)
has-operand1 5
has-operand2 6
Result:
30

Achieve Goal Completed
Ontology: wsmo-test
Goal Type: multiply-goal
has-operand1 5
has-operand2 6
Result:
30

Received request (Multiply):
has-operand1 5
has-operand2 6
Sending result:
30
SWS Creation & Usage Steps

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

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

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

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

• Publish against web service description

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

- Each WS has a mediator for used-mediator slot of capability
  - Some WS may share a mediator
- Define a kappa expression for assumption slot of WS capability
- Kappa expression format
  - \((\text{kappa} \ (\text{goal}) \ <\text{ocml relations}>)\)
- Getting the value of an input role
  - \((\text{wsmo-role-value} \ ?\text{goal} \ <\text{role-name}>)\)
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
Goal Based Invocation

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

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

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

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

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

Invocation
Invoke selected web service
European-exchange-rate
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))))
Tips

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