Semantic Web Services
Tutorial

Michael Stollberg and Armin Haller
DERI – Digital Enterprise Research Institute

3rd International Conference on Web Services (ICWS 2005)
Orlando, Florida, 2005 July 11
Agenda

Part I: Introduction to Semantic Web Services
  – Vision of Next Generation Web Technology
  – Semantic Web Service Challenges

Part II: The Web Service Modeling Ontology WSMO
  – Aims & Design Principles
  – Top Level Element Definitions

BREAK

Part III: A Walkthru Example
  – Virtual Travel Agency Example
  – Roles, Elements, Semantic Web Service technology usage

Part IV: The Web Service Execution Environment WSMX
  – Aims & Design Principles
  – Architecture & Components
PART I:

Introduction to Semantic Web Services

• The vision of the Semantic Web
• Ontologies as the basic building block
• Current Web Service Technologies
• Vision and Challenges for Semantic Web Services
The Vision

- 500 million users
- more than 3 billion pages

Static

WWW
URI, HTML, HTTP
The Vision

Serious Problems in

- information finding,
- information extracting,
- information representing,
- information interpreting and
- information maintaining.

Static

WWW
URI, HTML, HTTP

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

Dynamic

Web Services
UDDI, WSDL, SOAP

Static

WWW
URI, HTML, HTTP

Semantic Web
RDF, RDF(S), OWL

Bringing the computer back as a device for computation
The Vision

Bringing the web to its full potential

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 basic building block
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
coherecy description between Concepts / Properties / Relations via logical expressions

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

To make the Semantic Web working we need:

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

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

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

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

• and … Applications
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
The Promise of Web Services

*web-based SOA as new system design paradigm*
WSDL

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

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

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

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

- Simple Object Access Protocol
- W3C Recommendation

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

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

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:
The Web Service Modeling Ontology WSMO

- Aims & Working Groups
- Design Principles
- Top Level Notions
  - Ontologies
  - Web Services
  - Goals
  - Mediators
- Comparison to OWL-S
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)

- derived from and based on the Web Service Modeling Framework WSMF

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

- A Conceptual Model for SWS
- A Formal Language for WSMO
- A Rule-based Language for SWS
- Execution Environment for WSMO

3rd Internation Conference on Web Services (ISWC 2005), Orlando, Florida (USA), July 2005
WSMO Design Principles

• Web Compliance
• Ontology-Based
• Goal-driven
• Strict Decoupling
• Centrality of Mediation
• 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

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

every WSMO element 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
- Connectors between components with mediation facilities for handling heterogeneities

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

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

Objectives that a client wants to achieve by using Web Services

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
WSMO Web Service Description

Choreography --- Service Interfaces --- Orchestration

- complete item description
- quality aspects
- Web Service Management

Non-functional Properties

- Advertising of Web Service
- Support for WS Discovery

Capability

DC + QoS + Version + financial

Web Service Implementation

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

functional description

client-service interaction interface for consuming WS
- External Visible Behavior
- Communication Structure
- ‘Grounding’
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)
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 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**
  - executable 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
WSMO Web Service Interfaces

- service interfaces are concerned with service consumption and interaction
- Choreography and Orchestration as sub-concepts of Service Interface
- common requirements for service interface description:
  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 operations on them.
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 $GT(\omega)$:
  – state transition
  – general structure: \textbf{if} (condition) \textbf{then} (action)
  – different for Choreography and Orchestration
  – additional constructs: add, delete, update
Service Interface Example

**Communication Behavior of a Web Service**

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

State $\omega_1$

- a $\text{memberOf} A$ [att1 $\text{hasValue} x$
  
antt2 $\text{hasValue} y$
]

Received ontology instance $a$

Guarded Transition $GT(\omega_1)$

IF (a $\text{memberOf} A$ [att1 $\text{hasValue} x$])
THEN
(b $\text{memberOf} B$ [att2 $\text{hasValue} m$])

State $\omega_2$

- a $\text{memberOf} A$ [att1 $\text{hasValue} x,$
  
antt2 $\text{hasValue} y$
]
- b $\text{memberOf} B$ [att2 $\text{hasValue} m$
]

Sent ontology instance $b$
Future Directions

**Choreography:**
- interaction of services / service and client
- a „choreography interface“ describes the behavior of a Web Service for client-service interaction for consuming the service

**Orchestration:**
- how the functionality of a Web Service is achieved by aggregating other Web Services
- extends Choreography descriptions by control & data flow constructs between orchestrating WS and orchestrated WSs.

**Conceptual models**

**User language**
- based on UML2 activity diagrams
- graphical Tool for Editing & Browsing Service Interface Description

**Workflow constructs as basis for describing service interfaces:**
- workflow based process models for describing behavior
- on basis of generic workflow constructs (e.g. van der Aalst)

**Formal description of service interfaces:**
- ASM-based approach
- allows reasoning & mediation

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

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

• **Goal-driven Approach**, derived from AI rational agent approach
  - requester formulates objective independently
  - ‘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 (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*: 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 …**
  – 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** (WSMF):
  1. **Data Level:** mediate heterogeneous *Data Sources*
  2. **Protocol Level:** mediate heterogeneous *Communication Patterns*
  3. **Process Level:** mediate heterogeneous *Business Processes*
WSMO Mediators Overview
Mediator Structure

WSMO Mediator

- uses a Mediation Service via

1..n

Source Component

1

Target Component

- as a Goal
- directly
- optionally incl. Mediation

Mediation Services

Source Component
OO Mediator - Example

Merging 2 ontologies

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

Train Connection Ontology (s1)

Purchase Ontology (s2)

OO Mediator Mediation Service

Train Ticket Purchase Ontology

Discovery

Mediation Services
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
Comparison to OWL-S

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

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

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

• OWL-S is an ontology and a language to describe Web services
  – Strong relation to Web Services standards
    • rather than proposing another WS standard, OWL-S aims at enriching existing standards
    • OWL-S is grounded in WSDL and it has been mapped into UDDI
  – Based on the Semantic Web
    • Ontologies provide conceptual framework to describe the domain of Web services and an inference engine to reason about the domain
    • Ontologies are essential elements of interoperation between Web services

• WSMO is a conceptual model for the core elements of Semantic Web Services
  – core elements: Ontologies, Web Services, Goals, Mediators
    • language for semantic element description (WSML)
    • reference implementation (WSMX)
  – Mediation as a key element
  – Ontologies as data model
    • every resource description is based on ontologies
    • every data element interchanged is an ontology instance
OWL-S and WSMO

OWL-S profile ≈ WSMO capability + goal + non-functional properties

• OWL-S uses Profiles to express existing capabilities (advertisements) and desired capabilities (requests)
• WSMO separates provider (capabilities) and requester points of view (goals)
OWL-S and WSMO

OWL-S Process Model ≈ WSMO Service Interfaces

- **Perspective:**
  - OWL-S Process Model describes operations performed by Web Service, including consumption as well as aggregation
  - WSMO separates Choreography and Orchestration

- **Formal Model:**
  - OWL-S formal semantics has been developed in very different frameworks such as Situation Calculus, Petri Nets, Pi-calculus
  - WSMO service interface description model with ASM-based formal semantics
  - OWL-S Process Model is extended by SWRL / FLOWS

both approaches are not finalized yet
OWL-S and WSMO

OWL-S Grounding ≈ current WSMO Grounding

• OWL-S provides default mapping to WSDL
  – clear separation between WS description and interface implementation
  – other mappings could be used

• WSMO also defines a mapping to WSDL, but aims at an ontology-based grounding
  – avoid loss of ontological descriptions throughout service usage process
  – ‘Triple-Spaced Computing’ as innovative communication technology
Mediation in OWL-S and WSMO

• OWL-S does not have an explicit notion of mediator
  – Mediation is a by-product of the orchestration process
    • E.g. protocol mismatches are resolved by constructing a plan that
      coordinates the activity of the Web services
    – …or it results from translation axioms that are available to the
      Web services
      • It is not the mission of OWL-S to generate these axioms
  • WSMO regards mediators as key conceptual elements
    – Different kinds of mediators:
      • OO Mediators for ensuring semantic interoperability
      • GG, WG mediators to link Goals and Web Services
      • WW Mediators to establish service interoperability
    – Reusable mediators
    – Mediation techniques under development
Semantic Representation

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

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

  – WSMO is based on WSML a family of languages with a common basis for compatibility and extensions in the direction of Description Logics and Logic Programming
OWL and WSML

- WSML aims at overcoming deficiencies of OWL
- Relation between WSML and OWL+SWRL to be completed
## Summary

<table>
<thead>
<tr>
<th>Category</th>
<th>OWL-S</th>
<th>WSMO</th>
<th>current Web Service technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discovery</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Detection of suitable WS</em></td>
<td>Profile</td>
<td>Goals and Web Services (capability)</td>
<td>UDDI API</td>
</tr>
<tr>
<td><strong>Consumption &amp; Interaction</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>How to consume &amp; aggregate</em></td>
<td>Process Model</td>
<td>Service Interfaces (Choreography + Orchestration)</td>
<td>BPEL4WS / WS-CDL</td>
</tr>
<tr>
<td><strong>Invocation</strong></td>
<td>Grounding+ WSDL/SOAP</td>
<td>Grounding (WSDL / SOAP, ontology-based)</td>
<td>WSDL / SOAP</td>
</tr>
<tr>
<td><em>How to invoke</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mediation</strong></td>
<td></td>
<td>Mediators</td>
<td></td>
</tr>
<tr>
<td><em>Heterogeneity handling</em></td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PART III:

A Walkthru Example
Virtual Travel Agency Use Case

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

=> how does the interplay of James, VTA, and other Web Services look like?
Goal Description

- “book flight and hotel for the ICWS 2005 for James”
- goal capability postcondition: get a trip reservation for this

```
goal _"http://www.wsmo.org/examples/goals/icws2005"
importsOntology {"http://www.wsmo.org/ontologies/tripReservationOntology", ...}
capability
  postcondition
    definedBy
      ?tripReservation memberOf tr#reservation[
        customer hasValue fof#james,
        origin hasValue loc#innsbruck,
        destination hasValue loc#orlando,
        travel hasValue ?flight,
        accommodation hasValue ?conferenceHotel
        payment hasValue tr#creditcard
      ] and
      ?flight[airline hasValue tr#staralliance] memberOf tr#flight and
      ?hotel[name hasValue “Sheraton Safari Hotel”] memberOf tr#hotel.
```
VTA Service Description

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

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

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

- capability description (post-state)

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

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

James has Objective: “book a flight and a hotel for me for the ICWS 2005.”

Goal definition

Service Registry searches WS Discoverer result set includes VTA
Semantic Web Service Discovery

*find appropriate Web Service for automatically resolving a goal as the objective of a requester*

- **Aims:**
  - high precision discovery
  - maximal automation
  - effective discoverer architectures

- **Requirements:**
  - infrastructure that allows storage and retrieval of information about Web services
  - description of Web services functionality
  - description of requests or goals
  - algorithms for matching requesters for capabilities with the corresponding providers
Discovery Techniques

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

**Key Word Matching**
match natural language key words in resource descriptions

**Controlled Vocabulary**
ontology-based key word matching

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

Exact Match:
\[ \forall x. (G(x) \iff WS(x)) \]

PlugIn Match:
\[ \forall x. (G(x) \implies WS(x)) \]

Subsumption Match:
\[ \forall x. (G(x) \subseteq WS(x)) \]

Intersection Match:
\[ \exists x. (G(x) \land WS(x)) \]

Non Match:
\[ \exists x. (G(x) \land WS(x)) \]


3rd International Conference on Web Services (ISWC 2005), Orlando, Florida (USA), July 2005
Discovery Approach

- Matchmaking Notion to be used defined for each goal capability element
- Basic Procedure:

![Diagram showing the Discovery Approach process with decision paths for valid pre-state and post-state, leading to a match or abort conditions.]
Discoverer Architecture

- Discovery as central Semantic Web Services technology
- Integrated Discoverer Architectures admired:

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

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

  - usable Web Service

  - retrieve Service Descriptions
  - invoke Web Service

3rd International Conference on Web Services (ISWC 2005), Orlando, Florida (USA), July 2005
Service Interfaces

**Goal**
- Requested Capability
  - book flight & hotel
- Requested Interface
  1) send request
  2) select from offer
  3) receive confirmation

**Behavior Interface**: how entity can interact

**VTA**
- **Capability**
  - Interface (Chor.)
    1) get request
    2) provide offer
    3) receive selection
    4) send confirmation
  - VTA WS
  - Trip Booking

**Choreography**: interaction between entities

**Orchestration**: service aggregation for realizing functionality

**Flight WS**
- Capability
  - Interface (Chor.)
    1) flight request
    2) hotel request
    3) book flight
    4) book hotel

**Hotel WS**
- Capability
  - Interface (Chor.)
    1) get request
    2) provide offer
    3) receive selection
    4) send confirmation

3rd Internation Conference on Web Services (ISWC 2005), Orlando, Florida (USA), July 2005
VTA Service Description

• Behavior Interface
• Transition “get request” to “provide offer”

```plaintext
choreography VTABehaviorInterface
importsOntology {"http://www.wsmo.org/ontologies/tripReservationOntology“, …}
vocabularyIn {reservationRequest, …}
vocabularyOut {reservation, …}
guardedTransitions VTABehaviorInterfaceTransitionRules
if (reservationRequest memberOf tr#reservationRequest[ reservationItem hasValue tr#trip, origin hasValue loc#city, destination hasValue loc#city, passenger hasValue tr#passenger] then reservationOffer memberOf tr#reservation[ reservationItem hasValue tr#trip, reservationHolder hasValue ?reservationHolder] .
```
Choreography Discovery

**Goal**

**Requested Capability**
book flight & hotel

**Requested Interface**
1) send request
2) select from offer
3) receive confirmation

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

**VTA WS**
‘Trip Booking’

**Interface (Orch.)**
1) flight request
2) hotel request
3) book flight
4) book hotel

**Capability**
Flight WS

**Capability**
Hotel WS

- both behavior interfaces given (“static”)
- correct & complete consumption of VTA
=> existence of a valid choreography?

- VTA Orchestration & Behavior 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 algorithm & support from WSMO model
WSMO Service Interface Description Model

• common formal model for Service Interface description
  – ontologies as data model
  – based on ASMs
  – not restricted to any executable communication technology

• general structure:
  – 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
    • additional constructs: add, delete, update
Service Interface Example

Behavior Interface of a Web Service

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

State $\omega_1$

a memberOf A [att1 hasValue x, att2 hasValue y]

Guarded Transition $GT(\omega_1)$

IF (a memberOf A [att1 hasValue x])
THEN
(b memberOf B [att2 hasValue m])

State $\omega_2$

a memberOf A [att1 hasValue x, att2 hasValue y]

b memberOf B [att2 hasValue m]

received ontology instance a

sent ontology instance b

evolving ontology instance store

defined

$\Omega_{in}$ hasValues {
concept A [{att1 ofType X
att2 ofType Y} ...

$\Omega_{out}$ hasValues {
concept B [{att1 ofType W
att2 ofType Z} ...

3rd Internation Conference on Web Services (ISWC 2005), Orlando, Florida (USA), July 2005
Choreography Discovery

- a valid choreography exists if:
  - 1) **Information Compatibility**
    - compatible vocabulary
    - homogeneous ontologies
  - 2) **Communication Compatibility**
    - start state for interaction
    - a termination state can be reached without any additional input
Information Compatibility

If choreography participants have compatible vocabulary definitions:
- \( \Omega_{\text{in}}(S1) \) and \( \Omega_{\text{shared}}(S1) = \Omega_{\text{out}}(S2) \) and \( \Omega_{\text{shared}}(S2) \)
- determinable by Intersection Match from Discovery

\( \text{SI}_{S1}, \text{SI}_{S2}, O, M \models \exists x. (\Omega_{\text{S1 (in U shared)}}(x) \land \Omega_{\text{S2 (out U shared)}}(x)) \)
- more complex for multi-party choreographies

Prerequisite: choreography participants use homogeneous ontologies:
- semanticInteroperability\((S1, S2, ..., Sn)\)
- same ontologies in Service Interfaces, or usage of respective OO Mediators
Communication Compatibility

- **Definitions** (for “binary choreography” (only 2 services), more complex for multi-party choreographies)

**Valid Choreography State:**
\[ \omega_x(C(S1, S2)) \text{ if } \text{informationCompatibility}(\Omega S1(\omega_x), \Omega S2(\omega_x)) \]
- means: action in GT of S1 for reaching state \( \omega_x(S1) \) satisfies condition in GT of S2 for reaching state \( \omega_x(S2) \), or vice versa

**Start State:**
\[ \omega_\emptyset(C(S1, S2)) \text{ if } \Omega S1(\omega_\emptyset)=\emptyset \text{ and } \Omega S2(\omega_\emptyset)=\emptyset \text{ and } \exists \omega_1(C(S1, S2)) \]
- means: if initial states for choreography participants given (empty ontology, i.e. no information interchange has happened), and there is a valid choreography state for commencing the interaction

**Termination State:**
\[ \omega_T(C(S1, S2)) \text{ if } \Omega S1(\omega_T)=\text{noAction} \text{ and } \Omega S2(\omega_T)=\text{noAction} \text{ and } \exists \omega_T(C(S1, S2)) \]
- means: there exist termination states for choreography participants (no action for transition to next state), and this is reachable by a sequence of valid choreography states

- Communication Compatibility given if there exists a start state and a termination state is reachable without additional input by a sequence of valid choreography states
Communication Compatibility Example

James’ Goal Behavior Interface

$$\Omega_{S_1}(\omega \emptyset) = \{\emptyset\}$$

if $\emptyset$ then request

$$\Omega_{S_1}(\omega 1) = \{\text{request(out)}\}$$

if cnd1(offer) then changeReq

$$\Omega_{S_1}(\omega 2a) = \{\text{offer(in)}, \text{changeReq(out)}\}$$

if cnd2(offer) then order

$$\Omega_{S_1}(\omega 2b) = \{\text{offer(in)}, \text{order(out)}\}$$

if conf then $\emptyset$

$$\Omega_{S_1}(\omega 3) = \{\text{offer(in)}, \text{conf(in)}\}$$

VTA Behavior Interface

$$\Omega_{S_2}(\omega \emptyset) = \{\emptyset\}$$

if request then offer

$$\Omega_{S_2}(\omega 1) = \{\text{request(in), offer(out)}\}$$

if changeReq then offer

$$\Omega_{S_2}(\omega 2a) = \{\text{changeReq(in), offer(out)}\}$$

if order then conf

$$\Omega_{S_2}(\omega 2b) = \{\text{order(in), conf(out)}\}$$

existence of a valid Choreography
WW Mediators in Choreography

- if a choreography does not exist, then find an appropriate WW Mediator that
  - resolves possible mismatches to establish Information Compatibility (OO Mediator usage)
  - resolves process / protocol level mismatches in to establish Communication Compatibility
Orchestration

Control Structure for aggregation of other Web Services

- formally described service functionality decomposition
- only those aspects of WS realization wherefore other WS are aggregated
- aggregated WS used via their behavior interface
Orchestration Description & Validation

• Orchestration Description:
  – interaction behavior of “Orchestrator” with “orchestrated Web Services”
  – WSMO Service Interface description model, extension of Guarded Transitions general structure:
    \[
    \text{if condition then operation} \\
    \text{Operation} = (\text{Orchestrator}, \text{Web Service}, \text{Action})
    \]
  – Orchestrator serves as client for aggregated Web Services

• Orchestration Validation:
  – need to ensure that interactions with aggregated Web Service can be executed successfully
  => Choreography Discovery for all interaction of Orchestrator with each aggregated Web Service
Orchestration Validation Example

VTA Web Service Orchestration

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

Flight WS Behavior Interface

Start (VTA, FWS)
if request then offer
if order then confirmation

Termination (VTA, FWS)

Hotel WS Behavior Interface

Start (VTA, HWS)
if request then offer
if order then confirmation

Termination (VTA, HWS)

Orchestration is valid if valid choreography exists for interactions between Orchestrator and each aggregated Web Service, done by choreography discovery.
Service Composition and Orchestration

- Web Service Composition:
  - the realization of a Web Service by dynamically composing the functionalities of other Web Services
    - The new service is the **composite service**
    - The invoked services are the **component services**
  - a composite service can provide the skeleton for a Web Service (e.g. the VTA Web Service)

- Current Composition techniques only cover aspects for valid orchestrations partially
  - functional Web Service composition (on capability descriptions)
  - dynamic control and data flow construction for composite Web Service
  - delegation of client / goal behavior to component services

=> Orchestration Validation needed to ensure executable Web Service aggregations
Composition System Overview

(from Berardi, ESWC 2005 Semantic Web Services Tutorial)

functional requirements of the target service

non-functional requirements of the target service

specification of the process of the composite service

additional requirements for orchestration

Synthesis

Orchestration

Monitoring

service descriptions

available service 1

additional requirements for orchestration

client

target service invocation

available service invocation

available service n

Orchestration
Conclusions

• Semantic Web Service descriptions require
  – expertise in ontology & logical modeling
    => tool support for users & developers under development
  – understanding of Semantic Web Service technologies
    • what it does, and how it works
    • which are the related descriptive information

• Semantic Web Service technologies aim at automation of the Web Service usage process
  – users only define goal with tool support
  – ‘intelligent’ SWS middleware for automated Web Service usage

• state of the art in technology & tool development
  – theoretical approaches are converging; standardization efforts
  – prototypical SWS technologies existent
  – industrial strength SWS technology suites aspired in upcoming efforts
PART IV: The Web Service Execution Environment WSMX

• Aims & Design Principles
• WSMX Development Process and Releases
• Components and System Architecture
  – Components
  – Event-based Implementation
  – System Entry Points
  – Execution Semantics
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 data mediation (if required)
  – make the service invocation
• is based on the conceptual model provided by WSMO
• has formal execution semantics
• SO and event-based architecture based on microkernel design using technologies as J2EE, Hibernate, Spring, JMX, etc.
Design Principles

Strong Decoupling & Strong Mediation
autonomous components with mediators for interoperability

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

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

WSMO Design Principles == WSMX Design Principles
== SOA Design Principles
WSMX Usage Scenario
Development Process & Releases

• The development process for WSMX includes:
  – Establishing its conceptual model
  – Defining its execution semantics
  – Develop the architecture
  – Design the software
  – Building a working implementation

• Planned releases:

  - November 2004 (WSMX 0.1.5)
  - January 2005 (WSMX 0.2)
  - June 2005 (WSMX 0.3)
  - November 2005 (WSMX 0.4)

current status of components
Components & System Architecture

WSMT – Web Services Modelling Toolkit

- WSMX Management
- WSMX Monitor
- WSML Editor
- Choreography Editor
- Mediator Editor

WSMX Manager

WSMX Manager Core

- CM Wrapper
- RM Wrapper
- Parser Wrapper
- Discovery Wrapper
- Selector Wrapper
- DM Wrapper
- PM Wrapper
- Choreography Wrapper

- Interface
- Interface
- Interface
- Interface
- Interface
- Interface
- Interface

WSMX

Adminstration Framework Interface

- Communication Manager
- Resource Manager
- Parser
- Discovery
- Selector
- Data Mediator
- Process Mediator
- Choreography

Data and Communication Protocols Adapters

- Adapter 1
- Adapter 2
- Adapter n

Service Requesters

- Service Requesters
- Back-End Application
- Agent acting on behalf of service requester

Service Providers

- Web Service 1
- Web Service 2
- ...
- Web Service p

Grounding

CM Wrapper

Adapter Interface

Invocation

WSMO Objects

Non WSMO

Reasoner Interface

Reasoner

Component Wrapper

Interface

New Component

WSML Editor

WSMX Monitor

Choreography Editor

WSMX Management

WSMT – Web Services Modelling Toolkit
Selected Components

- Adapters
- Parser
- Invoker
- Choreography & Process Mediator
- Matchmaker
- Data Mediator
- Resource Manager
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 1.0 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)
Invoker

- WSMX V0.1 used the SOAP implementation from Apache AXIS
- Web Service interfaces were provided to WSMX as WSDL
- Both RPC and Document style invocations possible
- Input parameters for the Web Services were translated from WSML to XML using an additional XML Converter component.
Choreography & 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
- Choreography Engine & Process Mediator provides the means for runtime analyses of two choreography instances and uses mediators to compensate possible mismatches
Matchmaker

• responsible for finding appropriate Web Services to achieve a goal (discovery)
• currently the built-in matchmaking is performed by simple string-based matching; advanced semantic discoverers in prototypical stage
OOMediator

- 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)
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
Event-based Implementation

Core – Manager

“Business” Process – Internal Workflow

Event and Notification Distribution/Delivery Mechanism

- Choreography Wrapper
- Discovery Wrapper
- Mediator Interface
- Data Mediator Wrapper
- Communication Manager Wrapper

Choreography
Discovery
Mediator
Communication Manager
System entry points

• `storeEntity(WSMOEntity):Confirmation`  
  - provides an administration interface for storing any WSMO-related entities (Web Services, Goals, Ontologies)

• `realizeGoal(Goal, OntologyInstance):Confirmation`  
  - service requester expects WSMX to discover and invoke Web Service without exchanging additional messages

• `receiveGoal(Goal, OntologyInstance, Preferences):WebService[]`  
  - list of Web Services is created for given Goal  
  - requester can specify the number of Web Services to be returned

• `receiveMessage(OntologyInstance,WebServiceID, ChoreographyID):ChoreographyID`  
  - back-and-forth conversation to provide all necessary data for invocation  
  - involves execution of choreographies and process mediation between service interfaces
System Entry Points

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

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

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

receiveMessage
- Service Requester
- Adapter
- Communication Manager (Provider Side)
- Service Provider
- Data Mediator
- SOAP
Execution Semantics

Request to discover Web services.

WSMT – Web Services Modelling Toolkit

WSMX Manager

WSMX Manager Core

CM Wrapper
RM Wrapper
Parser Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper

Invoker Receiver

Resource Manager Interface

WSMO Objects

Non WSMO

Reasoner Interface

Reasoner

Component Wrapper

Interface

New Component

Service Providers

Web Service 1

Web Service 2

Web Service p

Service Requesters

Back-End Application

Agent acting on behalf of service requester

Data and Communication Protocols Adapters

Adapter 1
Adapter 2
... Adapter n

WSMO Objects
Non WSMO
Reasoner
Component Wrapper
Interface
New Component

3rd Internation Conference on Web Services (ISWC 2005), Orlando, Florida (USA), July 2005
Execution Semantics

Goal expressed in WSML is sent to WSMX System Interface

Service Requesters

Service Providers

WSMX System Interface

WSMX Manager

WSMX Manager Core

WSMT – Web Services Modelling Toolkit

CM Wrapper
RM Wrapper
Parser Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper

Interface
Interface
Interface
Interface
Interface
Interface
Interface
Interface

Invoker
Receiver
Grounding

Communication Manager
Resource Manager
Parser
Discovery
Selector
Data Mediator
Process Mediator
Choreography

Resource Manager Interface

WSMO Objects
Non WSMO

Reasoner Interface

Reasoner

Component Wrapper
New Component

Component Wrapper

Interface
Execution Semantics
Execution Semantics

WSMT – Web Services Modelling Toolkit

WSMX Manager

WSMX Manager Core

Component Wrapper

Interface

New Component

Resource Manager Interface

WSMO Objects

Non WSMO

Reasoner Interface

Reasoner

Com. M. informs Core that Goal has been received

CM Wrapper

RM Wraper

Parser

Discovery Wrapper

Selector Wraper

DM Wraper

PM Wraper

Choreography Wraper

Invoker

Receiver

Grounding

Interface

Interface

Interface

Interface

Interface

Interface

Interface

Interface

Service Requesters

Service Providers

Data and Communication Protocols Adapters

WSML Editor

WSMX Monitor

WSMX Managment

Choreography Editor

Mediator Editor

Web Service 1

Web Service 2

Web Service p

Back-End Application

Agent acting on behalf of service requester

Agent acting on behalf of service requester

Agent acting on behalf of service requester
Execution Semantics

WSMX – Web Services Modelling Toolkit

WSMX Management
WSMX Monitor
WSML Editor
Choreography Editor
Mediator Editor

WSMX Manager
WSMX Manager Core

Chor. wrapper picks up event for Chor. component

CM Wrapper
RM Wrapper
Parser Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper

Component Wrapper
Interface
Non WSMO
Reasoner Interface
Non WSMO
Reasoner

Resource Manager Interface
WSMO Objects
Reasoner

Service Requesters
Back-End Application
Agent acting on behalf of service requester

Data and Communication Protocols Adapters
Adapter 1
Adapter 2
Adapter n...

Service Providers
Web Service 1
Web Service 2
... Web Service p

WSML Editor
WSMX Monitor
Choreography Editor
WSMX Management
WSMX Monitor
WSML Editor
Choreography Editor
Mediator Editor

WSMO Objects
Reasoner
Chor. wrapper picks up event for Chor. component

Execution Semantics
Execution Semantics
Execution Semantics
Execution Semantics

WSMT – Web Services Modelling Toolkit

- WSMX Managment
- WSMX Monitor
- WSML Editor
- Choreography Editor
- Mediator Editor

WSMX Manager Core

- CM Wrapper
- RM Wrapper
- Parser Wrapper
- Discovery Wrapper
- Selector Wrapper
- DM Wrapper
- PM Wrapper
- Choreography Wrapper

WSMX Manager

- Administration Framework Interface

WSMX System Interface

- CM Wrapper
- RM Wrapper
- Discovery Wrapper
- Selector Wrapper
- Data Mediator
- Process Mediator
- Choreography

Component Wrapper

WSMGO Objects

Non WSMO

Reasoner

Reasoner Interface

Reasoner

Component Wrapper

Interface

New Component

WSML – Web Services Modelling Toolkit

Service Requesters

- Service Requesters
  - Back-End Application
  - Agent acting on behalf of service requester

Service Providers

- Service Providers
  - Web Service 1
  - Web Service 2
  - ... Web Service p

Agent

Data and Communication Protocols Adapters

- Adapter 1
- Adapter 2
- ... Adapter n

WSML Editor

WSMX Monitor

Choreography Editor

WSMX Management

WSML goal is parsed to internal format.
Execution Semantics

WSMT – Web Services Modelling Toolkit

WSMX Management  WSMX Monitor  WSML Editor  Choreography Editor  Mediator Editor

WSMX

WSMX Manager

WSMX Manager Core

CM Wrapper  RM Wrapper  Parser Wrapper  Discovery Wrapper  Selector Wrapper  DM Wrapper  PM Wrapper  Choreography Wrapper

Interface  Interface  Interface  Interface  Interface  Interface  Interface  Interface

Communication Manager  Resource Manager  Parser  Discovery  Selector  DM  PM  Choreography

Invoker  Receiver  Invoker  Receiver  Invoker  Receiver  Invoker  Receiver

Discovery is invoked for parsed goal.

Resource Manager Interface

WSMO Objects  Non WSMO

Reasoner Interface

Reasoner  New Component

Component Wrapper

Invocation Semantics

Service Requesters

Service Providers

WSML Editor  WSMX Monitor  Choreography Editor  WSMX Management

Web Service 1  Web Service 2  ...  Web Service p

Back-End Application  Agent acting on behalf of service requester

Adapter 1  Adapter 2  ...  Adapter n

Invoking Discovery for parsed goal.
Execution Semantics
After data mediation, Discovery iterates, if needed through last steps until result set is finished.
Execution Semantics

WSMX System Interface
- WSMX Manager
- WSMX Manager Core
- Administration Framework Interface
- Data and Communication Protocols Adapters
  - Adapter 1
  - Adapter 2
  - Adapter n...
- Grounding
- WSMO Objects
- Non WSMO
- Reasoner
- Resource Manager Interface
- Reasoner Interface
- WSMO Objects
- Non WSMO
- Reasoner
- Resource Manager Interface
- Reasoner Interface

WSMT – Web Services Modelling Toolkit
- WSMX Managment
- WSMX Monitor
- WSML Editor
- Choreography Editor
- Mediator Editor

Service Requesters
- Back-End Application
- Agent acting on behalf of service requester
- Adapter 1
- Adapter 2
- Adapter n...

Service Providers
- Web Service 1
- Web Service 2
- Web Service p

WSML Editor
- WSMT – Web Services Modelling Toolkit
- WSMX Monitor
- WSML Editor
- Choreography Editor
- Mediator Editor

Selection is invoked to relax result set to finally one service.

3rd Internation Conference on Web Services (ISWC 2005), Orlando, Florida (USA), July 2005
Execution Semantics
Execution Semantics
Execution Semantics

Set of Web Service descriptions expressed in WSML sent to adapter.
Execution Semantics

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

WSMX – Web Services Modelling Toolkit

WSMX Management
WSMX Monitor
WSML Editor
Choreography Editor
Mediation Editor

Administration Framework Interface

WSMX Manager

WSMX Manager Core

WSM – Web Services Modelling Toolkit

WSMT – Web Services Modelling Toolkit

Service Providers

Requesters

Back-End Application
Agent acting on behalf of service requester

Data and Communication Protocols Adapters

Adapter 1
Adapter 2
... Adapter n

Component Wrapper
Interface
New Component

Resource Manager Interface

WSMO Objects
Non WSMO
Reasoner Interface
Reasoner

Choreography

Execution Semantics

Set of Web Service descriptions expressed in requester’s own format returned to goal requester.
Conclusions

• Conceptual model is WSMO (with some add-ons)
• End to end functionality for executing SWS
• Has a formal execution semantics
• Real implementation
• Open source code base at SourceForge
• Event-driven component architecture
• Developers welcome
WSMX @ Sourceforge.net

Project: Web Services Execution Environment: Summary

Summary | Admin | Home Page | Forums | Tracker | Bugs | Support | Patches | RFE | Lists | Tasks | Docs | Screenshots | News | CVS | Files | Donations |

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

- Development Status: Alpha
- Intended Audience: Developers, Scientists/Researchers
- License: MIT License
- Programming Language: Java
- Topic: Distributed Computing

Project UNIX name: wsmx
Registered: 2004-06-29 12:46
Activity Percentile (last week): 37.66%
View project activity statistics
View list of RSS feeds available for this project

Latest File Releases

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
<th>Date</th>
<th>Notes / Monitor</th>
<th>Download</th>
</tr>
</thead>
<tbody>
<tr>
<td>toolkit</td>
<td>WSMT v0.1</td>
<td>March 16, 2005</td>
<td></td>
<td>Download</td>
</tr>
<tr>
<td>wsmx-components</td>
<td>WSMX Components 0.1.6</td>
<td>January 31, 2005</td>
<td></td>
<td>Download</td>
</tr>
<tr>
<td>wsmx-core</td>
<td>WSMX Core 0.01</td>
<td>July 30, 2004</td>
<td></td>
<td>Download</td>
</tr>
</tbody>
</table>

[View All Project Files]

Public Areas

- Project Home Page
- Supporters of this project/Make a donation

Latest News

- WSMT v0.1 Released
  - morgen - 2005-03-16 12:17
  - [Read More/Comment]

3rd Internation Conference on Web Services (ISWC 2005), Orlando, Florida (USA), July 2005
Closing, Outlook, Acknowledgements
Tutorial Wrap-up

• The targets of the presented tutorial were to:
  – understand aims & challenges within Semantic Web Services
  – understand Semantic Web Service Frameworks:
    • aims, design principles, and paradigms
    • ontology elements & description

• an overview of Semantic Web Service techniques:
  – element description
  – discovery
  – choreography and service interoperability determination
  – orchestration and composition

• present WSMX a future Web Service based IT middleware
  – design and architecture
  – components design

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

- Although WSMO (and OWL-S) are the main initiatives on Semantic Web services, they are not the only ones:

  - Semantic Web Services Interest Group
    - Interest group founded at W3C to discuss issues related to Semantic Web Services (http://www.w3.org/2002/ws/swsig/)
    - Standardization Working Group in starting phase

  - SWSI: International initiative to push toward a standardization of SWS (http://www.swsi.org)

- Semantic Web services are entering the main stream
  - UDDI is adopting OWL for semantic search
  - WSDL 2 will contain a mapping to RDF
  - The use of semantics is also discussed in the context of standards for WS Policies
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.

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