State of Affairs in Semantic Web Services

Canadian Semantic Web Symposium 2006

Michael Stollberg
Digital Enterprise Research Institute DERI
University of Innsbruck, Austria
PART I:

Introduction to Semantic Web Services
"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 Vision

- 500 million users
- more than 3 billion pages

Static

WWW
URI, HTML, HTTP
The Vision

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

Static
WWW
URI, HTML, HTTP

Semantic Web
RDF, RDF(S), OWL

The Vision

Dynamic
Web Services
UDDI, WSDL, SOAP

Enable Computing over the Web

Static
WWW
URI, HTML, HTTP

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

Automated Web Service Usage

Dynamic

Web Services
UDDI, WSDL, SOAP

Semantic Web Services

Static

WWW
URI, HTML, HTTP

Semantic Web
RDF, RDF(S), OWL

The Semantic Web

• the next generation of the WWW

• information has machine-processable and machine-understandable semantics

• not a separate Web but an augmentation of the current one

• ontologies as base technology
Ontology Definition

- **Ontology Definition**
  - **Concept**: conceptual entity of the domain
  - **Property**: attribute describing a concept
  - **Relation**: relationship between concepts or properties
  - **Axiom**: coherency description between Concepts / Properties / Relations via logical expressions
  - **Instance**: individual in the domain

- **Ontology Example**
  - **Concept**: Person, Student, Professor, Lecture
  - **Property**: name, email, research field, topic, student ID, lecture no.
  - **Relation**: isA, memberOf, attends, holds
  - **Axiom**:
    - holds(Professor, Lecture) => Lecture.topic = Professor.researchField
    - Ann memberOf student name = Ann Lee studentID = 12345
Ontology Technology

To make the Semantic Web working we need:

- **Ontology Languages:**
  - expressivity
  - reasoning support
  - web compliance

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

- **Ontology Management Techniques:**
  - (collaborative) editing and browsing
  - storage and retrieval
  - versioning and evolution Support

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

Web Services

- loosely coupled, reusable components
- encapsulate discrete functionality
- distributed
- programmatically accessible over standard internet protocols
- add new level of functionality on top of the current web

=> base technology for service oriented architectures (SOA) on the Web
The Promise of Web Services

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

- **UDDI Registry**: Points to Description
  - Finds Service
- **WSDL**: Points to Service
  - Describes Service
- **Service Consumer**
  - SOAP: Communicates with XML Messages

---

**WSDL**

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

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

- Simple Object Access Protocol
- W3C Recommendation

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

UDDI

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

Registry for Web Services:
- provider
- service information
- technical access
Deficiencies of WS Technology

• current technologies allow usage of Web Services
• but:
  – only syntactical information descriptions
  – syntactic support for discovery, composition and execution
  => Web Service usability, usage, and integration needs to be inspected manually
  – no semantically marked up content / services
  – no support for the Semantic Web

=> current Web Service Technology Stack failed to realize the promise of Web Services

Semantic Web Services

Semantic Web Technology
• allow machine supported data interpretation
• ontologies as data model

=> Semantic Web Services as integrated solution for realizing the vision of the next generation of the Web
Semantic Web Services

- define exhaustive description frameworks for describing Web Services and related aspects (Web Service Description Ontologies)
- support ontologies as underlying data model to allow machine supported Web data interpretation (Semantic Web aspect)
- define semantically driven technologies for automation of the Web Service usage process (Web Service aspect)

Web Service Usage Process

1. Deployment create & publish Web service description
2. Discovery determine usable services for a request
3. Composition combine services to achieve a goal
4. Selection choose most appropriate service among the available ones
5. Mediation solve mismatches (data, protocol, process) that hamper interoperation
6. Execution invoke Web services following programmatic conventions
Web Service Execution Support

- **Monitoring**: control the execution process
- **Compensation**: provide transactional support and undo or mitigate unwanted effects
- **Replacement**: facilitate the substitution of services by equivalent ones
- **Auditing**: verify that service execution occurred in the expected way

PART II:

Semantic Web Service Frameworks
Aims and Requirements

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

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

Web Service Modeling Ontology WSMO

• Comprehensive Framework for SESA
  Semantically Empowered Service-Oriented Architecture
  – top level notions = SESA core elements
  – conceptual model & axiomatization
  – ontology & rule language

• International Consortium (mostly European)
  – started in 2004
  – 78 members from 20 organizations
  – W3C member submission in April 2005
WSMO Working Groups

Conceptual Model & Axiomatization for SWS

Formal Language for WSMO

Execution Environment for WSMO

Ontology & Rule Language for the Semantic Web

WSMO Top Level Notions

Objectives that a client wants to achieve by using Web Services

Formally specified terminology of the information used by all other components

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

Connectors between components with mediation facilities for handling heterogeneities

W3C submission 13 April 2005
Non-Functional Properties

relevant, non-functional aspects for WSMO elements

- Dublin Core Metadata Set:
  - complete item description
  - used for resource management
- Versioning Information
  - evolution support
- Quality of Service Information
  - availability, stability
- Other
  - owner, financial

Non-Functional Properties List

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

Objectives that a client wants to achieve by using Web Services

Formally specified terminology of the information used by all other components

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

Connectors between components with mediation facilities for handling heterogeneities

Ontology Usage & Principles

- **Ontologies are the ‘data model’ throughout WSMO**
  - all WSMO element descriptions rely on ontologies
  - all data interchanged in Web Service usage are ontologies
  - Semantic information processing & ontology reasoning

- **WSMO Ontology Language WSML**
  - conceptual syntax for describing WSMO elements
  - logical language for axiomatic expressions (WSML Layering)

- **WSMO Ontology Design**
  - Modularization: import / re-using ontologies, modular approach for ontology design
  - De-Coupling: heterogeneity handled by **OO Mediators**
Ontology Specification

- Non functional properties (see before)
- Imported Ontologies importing existing ontologies where no heterogeneities arise
- Used mediators OO Mediators (ontology import with terminology mismatch handling)

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

WSMO Web Services

Objectives that a client wants to achieve by using Web Services

Formally specified terminology of the information used by all other components

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  
- Web Service Management

Non-functional Properties

- Advertising of Web Service
- Support for WS Discovery

Capability

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

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

Choreography --- Service Interfaces --- Orchestration

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 data level mismatch resolution
  - WG Mediator: link to a Goal wherefore service is not usable a priori
- Shared Variables: scope is entire capability
- Pre-conditions
  what a web service expects in order to be able to provide its service. They define conditions over the input.
- Assumptions
  conditions on the state of the world that has to hold before the Web Service can be executed
- Post-conditions
  describes the result of the Web Service in relation to the input, and conditions on it
- Effects
  conditions on the state of the world that hold after execution of the Web Service (i.e. changes in the state of the world)
Example VTA Web Service

- Web service for booking tickets or complete trips
- WSMO capability precondition

<table>
<thead>
<tr>
<th>capability VTA capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>sharedVariables {?item, ?passenger, ?creditCard, ?initialBalance, ?reservationPrice}</td>
</tr>
<tr>
<td>precondition definedBy exists ?reservationRequest</td>
</tr>
<tr>
<td>( ?reservationRequest[ reservationItem hasValue ?item, passenger hasValue ?passenger, payment hasValue ?creditcard] memberOf tr#reservationRequest and ( ?item memberOf tr#trip or ?item memberOf tr#ticket) and ?passenger memberOf pr#person and ?creditCard memberOf po#creditCard and ( ?creditCard[type hasValue po#visa] or ?creditCard[type hasValue po#mastercard]) ).</td>
</tr>
</tbody>
</table>

Example VTA Web Service

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

<table>
<thead>
<tr>
<th>assumption definedBy</th>
</tr>
</thead>
<tbody>
<tr>
<td>po#validCreditCard(?creditCard) and</td>
</tr>
<tr>
<td>?creditCard[balance hasValue ?initialBalance] and</td>
</tr>
<tr>
<td>(?initialBalance &gt;= ?reservationPrice).</td>
</tr>
</tbody>
</table>
Example VTA Web Service

- capability description (post-state)

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

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

Choreography & Orchestration

- VTA example:

  ```plaintext
  Choreography = how to interact with the service to consume its functionality
  Orchestration = how service functionality is achieved by aggregating other Web Services
  ```
Choreography Interfaces

*interface for consuming Web Service*

- **External Visible Behavior**
  - those aspects of the workflow of a Web Service where Interaction is required
  - described by workflow constructs: sequence, split, loop, parallel
- **Communication Structure**
  - messages sent and received
  - their order (communicative behavior for service consumption)
- **Grounding**
  - executable communication technology for interaction
  - choreography related errors (e.g. input wrong, message timeout, etc.)
- **Formal Model**
  - reasoning on Web Service interfaces (service interoperability)
  - semantically enabled mediation on Web Service interfaces

Orchestration Aspects

*interface for interaction with aggregated Web Services*

- decomposition of service functionality
- other Web services consumed via their choreography interfaces
WSMO Web Service Interfaces

- behavior interfaces of Web services and clients for “peer-2-peer” interaction
- Choreography and Orchestration as sub-concepts of Service Interface with common description language
- Web Service Interface Description aspects:
  1. represent the dynamics of information interchange during service consumption and interaction
  2. support ontologies as the underlying data model
  3. appropriate communication technology for information interchange
  4. sound formal model / semantics of service interface specifications in order to allow advanced reasoning on them

Ontologized Abstract State Machines

- Vocabulary $\Omega$:
  - ontology schema(s) used in service interface description
  - usage for information interchange: in, out, shared, controlled
- States $\omega(\Omega)$:
  - a stable status in the information space
  - defined by attribute values of ontology instances
- Guarded Transition $\text{GT}(\omega)$:
  - state transition
  - general structure: if (condition) then (update)
    - condition on current state, update = changes in state transition
    - all $\text{GT}(\omega)$ whose condition is fulfilled fire in parallel
Example Hotel Web Service

- choreography interface (state signature)

```plaintext
interface htl#BookHotelInterface
collection
stateSignature
importsOntology htl#simpleHotelOntology
in
htl#HotelRequest withGrounding _"http://...",
htl#HotelConfirm withGrounding _"http://...",
htl#HotelCancel withGrounding _"http://..."
out
htl#HotelNotAvailable withGrounding _"http://...",
htl#HotelOffer withGrounding _"http://..."
shared
htl#Hotel,
htl#HotelAvailable,
htl#HotelBooked
```

Example Hotel Web Service

- choreography interface (transition rules)

```plaintext
ctl_state (htl#start, htl#offerMade, htl#noAvail, htl#confirmed, htl#cancelled)
transitionRules
if (ctl_state = htl#start) then
forall (?req, ?date, ?loc, ?client) with
?req[trv#date hasValue ?date, trv#location hasValue ?loc,
htl#client hasValue ?client] memberOf htl#HotelRequest do
add(htl#offer(?req)[trv#date hasValue ?date,
trv#hotelName hasValue ?name, trv#location hasValue ?loc,
htl#client hasValue ?client] memberOf htl#HotelOffer)
ctl_state := htl#offerMade
| |
add(htl#notAvailable(?req)[trv#date hasValue ?date,
trv#location hasValue ?loc] memberOf htl#HotelNotAvailable)
ctl_state := htl#noAvail
endForall
eendif
```
WSMO Goals

Objectives that a client wants to achieve by using Web Services

Formally specified terminology of the information used by all other components

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

Connectors between components with mediation facilities for handling heterogeneities

Goals

*client objective specification along with all information needed for automated resolution*

- **Goal-driven Approach**, derived from AI rational agent approach
  - ontological de-coupling of Requester and Provider
  - ‘intelligent’ mechanisms detect suitable services for solving the Goal
  - service re-use & knowledge-level client side support

- **Usage of Goals within Semantic Web Services**
  - A Requester (human or machine) defines a Goal to be resolved independently (i.e. subjectively) on the knowledge level
  - SWS techniques / systems automatically determine Web Services to be used for resolving the Goal (discovery, composition, execution, etc.)
  - Goal Resolution Management is realized in implementations
Goal-driven Architecture

Client-Side

- Goal
  - objective (desired final state)
  - input for service usage
  - goal resolution constraints, preferences, and policies

Goal Resolution Plan
- goal resolution algorithm
- decomposition (optional)
- service usage / invocation

corresponds to / creation of

Service-Side

- (Web) Service Implementation
- service detection & composition

Goal Model (WSMO 2.0)

Abstract Goal

- importOntology: ontology
- precondition: axiom
- effect: axiom
- resolutionConstraints: axiom

Composite Goal

- nonFunctionalProperties: nFP
- input: axiom
- output: axiom
- subGoal: atomic goal, composite goal
- controlFlow: controlFlow
- dataFlow: dataFlow

Atomic Goal

- role instantiation

Goal Instance

- nonFunctionalProperties: nFP
- input: axiom
- output: axiom
- clientInterface: interface
WSMO Mediators

Objectives that a client wants to achieve by using Web Services

- Formally specified terminology of the information used by all other components

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

Connectors between components with mediation facilities for handling heterogeneities

Mediation

- Heterogeneity ...
  - mismatches on structural / semantic / conceptual / level
  - occur between different components that shall interoperate
  - especially in distributed & open environments like the Internet

- Concept of Mediation (Wiederhold, 94):
  - Mediators as components that resolve mismatches
  - declarative approach:
    - semantic description of resources
    - ‘intelligent’ mechanisms that resolve mismatches independent of content
  - mediation cannot be fully automated (integration decision)

- Levels of Mediation within Semantic Web Services:
  1. Data Level: heterogeneous Data Sources
  2. Functional Level: heterogeneous Functionalities
Other Approaches

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

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

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

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

- Discussed here:
  - Central Features
  - Commonalities and Differences

---

**OWL-S**

Upper Ontology for Web Service Descriptions

- capability description (IOPE)
- non-functional properties
- usage: (1) WS advertisement, (2) WS request formulation
- specification of service access information
- builds upon WSDL to define message structure and physical binding layer
- specifies communication protocols & language, transport mechanisms, etc.
- describes internal processes of the service
- defines service interaction protocol for (a) consumption and (b) WS interaction
- process types: simple, atomic, composite
- specifies: (1) abstract messages (ontological content), (2) control flow constructs, (3) perform construct
OWL-S and WSMO

- **OWL-S** = ontology and language to describe Web services
- **WSMO** = ontology and language for core elements of Semantic Web Service systems

Main Description Elements Correlation:

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

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

**OWL-S Grounding** = current WSMO Grounding
- Goals and Mediators not in scope
- deficiencies in Service Model (process description model / language not adequate) => SWSF

---

SWSF

- Process Model for Web Services (FLOWS)
- although self-contained, commonly understood as extension of OWL-S / refinement of Service Model

<table>
<thead>
<tr>
<th>Module</th>
<th>Explanation</th>
<th>Major Concepts</th>
</tr>
</thead>
</table>
| FLOWS-Core      | basic notions of services as activities composed of atomic activities | Service-AtomicProcess
|                 |                                                       | supportedOr message channel |
| Control Constraints | common workflow-style process constructs, including OWL-S process model concepts. | Split, Sequence, Unordered Choice, Iterate, Terminate, RepeatUntil |
| Ordering Constraints | allowable sequences of activities defined by sequencing properties of atomic processes | OrderableActivity |
| Occurrence Constraints | support for nondeterministic activities within services | OccActivity |
| State Constraints | specify activities that are triggered by states (of an overall system) | TriggeredActivity |
| Exception Constraints | basic infrastructure for modeling exceptions | Exception |
Semantic Web Services, CSWWS 2006, Quebec (Canada), 06 June 2006

WSDL-S

Semantic annotation of WSDL descriptions
1. annotate XML Schema with domain ontology

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

2. pre-conditions & effects for operations

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

3. WS categorization by ontology-based keywords

```xml
<wssem:category name= "Electronics" taxonomyURI="http://www.naics.com/" taxonomyCode="443112"/>
```

Commonalities & Differences

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

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

- WSMO defines “core elements for SESA” while all others are only concerned with describing Web Services
PART III:

Semantic Techniques for Automated Web Service Usage
**SWS Challenges**

- Web services as loosely coupled components that shall interoperate dynamically and automatically
- Techniques required for:
  - **Discovery**
    - how are Web services found and selected?
  - **Composition**
    - how to aggregate Web Services into a complex functionality?
  - **Conversation**
    - how to ensure automated interaction of Web Services?
  - **Invocation**
    - how to access and invoke Semantic Web Services?
  - **Mediation**
    - how are data and protocol mismatches resolved?
- Systems for automated Web service usage:
  - resource editing and management
  - functional components
  - APIs, execution control, integrated & flexible architectures

---

**Web Service Usage Process**

[Diagram showing the process of web service usage with steps for request, discovery, composition, communication, and execution.]
Web Service Discovery

detect directly usable Web services out of available ones

- Discovery Techniques
  - **Key Word Matching**
    match natural language key words in resource descriptions
  - **Controlled Vocabulary**
    ontology-based key word matching
  - **Semantic Matchmaking**
    … what Semantic Web Services aim at

- Selection: choose most appropriate Web Service with respect to:
  - Quality of Service (security, robustness, availability)
  - context (regional, business / social communities)
  - preferences and policies
  - financial
  - …

Matchmaking Notions

**Exact Match:**
\[ G, WS, O, M \models \forall x. (G(x) \iff WS(x)) \]

**PlugIn Match:**
\[ G, WS, O, M \models \forall x. (G(x) \implies WS(x)) \]

**Subsumption Match:**
\[ G, WS, O, M \models \forall x. (G(x) \subseteq WS(x)) \]

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

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

Discovery Procedure

- goal-driven reasoning
- remarks:
  - precondition & assumption / postcondition & effect semantically the same
  - only situation that guarantees goal resolution by Web service usage is
    \[ \text{subsume} \preceq \text{pre}(G, WS) \quad \text{and} \quad \text{plugin} \succeq \text{eff}(G, WS) \]

Web Service Composition

combine several Web services for solving a request

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

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

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

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

• mainly AI Planning on functional descriptions

(a) \( \text{compCandidate}(Q, S) \Rightarrow \exists S_j \in \{S_1, \ldots, S_n\}. \phi_{\text{post}}^Q \land \phi_{\text{pre}}^{S_j} \land \forall S_j \in \{S_1, \ldots, S_n\}. \phi_{\text{post}}^{S_j} \land \forall S_j \in \{S_1, \ldots, S_n\}. \phi_{\text{pre}}^{S_j} \land \forall \phi_{\text{pre}}^{S_j} \land \forall \phi_{\text{post}}^{S_j} \)

(b) \( \text{compCandidate}(Q, S) \Rightarrow \forall S_j \in \{S_1, \ldots, S_n\}. \phi_{\text{post}}^{S_j} \land \forall S_j \in \{S_1, \ldots, S_n\}. \phi_{\text{pre}}^{S_j} \land \forall S_j \in \{S_1, \ldots, S_n\}. \phi_{\text{pre}}^{S_j} \land \forall \phi_{\text{pre}}^{S_j} \land \forall \phi_{\text{post}}^{S_j} \)

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

1. add \( S_j \) to composition \( Q \circ S \circ \ldots \circ S \)

2. create new request \( Q' \) such that \( \phi_{\text{post}}^{Q'} = \phi_{\text{pre}}^{Q} \) and \( \phi_{\text{pre}}^{Q'} = \phi_{\text{pre}}^{Q} \land \phi_{\text{pre}}^{S_j} \)

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

repeat until \( \phi_{\text{post}}^{Q'} = \phi_{\text{pre}}^{Q} \).

backward chaining (goal-driven reasoning)

• main technique: AI Planning

– set of Web services with control & data flow
– composition skeleton with all needed Web services

Semantic Web Services, CSWWS 2006. Quebec (Canada), 06 June 2006
Behavioral Composition

- does there exists an executable sequence for interaction wrt communication behavior of composed Web services?
  - analyze behavioral interfaces
  - determine existence of valid choreography
    “Choreography Discovery”
- techniques:
  - model-checking (for state-based descriptions)
  - conformance testing (for process-based)
- exponential time

Choreography Discovery

- both choreography interfaces given ("static")
- correct & complete consumption of VTA
  => existence of a valid choreography?
- VTA Orchestration & Chor. Interfaces of aggregated WS given
  => existence of a valid choreography between VTA and each aggregated WS?
- Choreography Discovery as a central reasoning task in Service Interfaces
  - ‘choreographies’ do not have to be described, only existence determination
Choreography Discovery

- A valid choreography exists if:
  1) **Signature Compatibility**
     - Homogeneous ontologies
     - Compatible in- and outputs
  2) **Behavior Compatibility**
     - Start state for interaction
     - A termination state can be reached without any additional input

**Behavior Compatibility Example**

<table>
<thead>
<tr>
<th>Goal Behavior Interface</th>
<th>VTA Behavior Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Omega_G(\omega_0) = {\emptyset}$</td>
<td>$\Omega_{VTA}(\omega_0) = {\emptyset}$</td>
</tr>
<tr>
<td>if $\emptyset$ then request</td>
<td>if $\text{request}$ then offer</td>
</tr>
<tr>
<td>$\Omega_G(\omega_1) = {\text{request(out)}}$</td>
<td>$\Omega_{VTA}(\omega_1) = {\text{request(in), offer(out)}}$</td>
</tr>
<tr>
<td>if $\text{cnd1(offer)}$ then changeReq</td>
<td>if $\text{changeReq}$ then offer</td>
</tr>
<tr>
<td>$\Omega_G(\omega_2a) = {\text{offer(in), changeReq(out)}}$</td>
<td>$\Omega_{VTA}(\omega_2a) = {\text{changeReq(in), offer(out)}}$</td>
</tr>
<tr>
<td>if $\text{cnd2(offer)}$ then order</td>
<td>if $\text{order}$ then conf</td>
</tr>
<tr>
<td>$\Omega_G(\omega_2b) = {\text{offer(in), order(out)}}$</td>
<td>$\Omega_{VTA}(\omega_2b) = {\text{order(in), conf(out)}}$</td>
</tr>
<tr>
<td>if $\text{conf then } \emptyset$</td>
<td></td>
</tr>
<tr>
<td>$\Omega_G(\omega_3) = {\text{offer(in), conf(in)}}$</td>
<td></td>
</tr>
</tbody>
</table>

Start $\omega_1(C)$

$\omega_2(C)$

$\omega_3(C)$

$\omega_4(C)$

Termination

Valid choreography existent
Orchestration Validation Example

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

Mediation

- **Heterogeneity ...**
  - mismatches on structural / semantic / conceptual / level
  - a natural characteristic of the world (constructivism)
  - hamper interoperability of resources / software components
  - occurs esp. in decentralized & open environments like the Internet

- **Concept of Mediation** (Wiederhold, 1994):
  - *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)

=> **Semantic Web Mediation Architecture** (WSMO Mediators):
1. identify heterogeneity types & develop techniques for handling these
2. mediation component specification & usage
3. execution facilities & software architecture
Heterogeneity Types

1. Protocol / Representation Formats
   – different representation languages
   – different communication protocols

2. Terminology & Knowledge
   – different ontologies
   – denotation < explication < conceptualization

3. Functional
   – provided and requested functionalities do not match precisely

4. Communication & Business Processes
   – different supported / required processes for WS consumption & interaction

Protocol Level Mediation

- Different Languages / Protocols
  => Adaptors for Transformation
  - syntactic transformation
  - mappings between language constructs

- Usage:
  - interoperability between systems with
    - different languages (e.g. OWL – WSML, etc.)
    - different communication protocols (EDI – RosettaNet)
  - grounding for Semantic Web services (lifting & lowering between syntactic and semantic level)
Data Level Mediation Techniques

- Ontology Integration
  - terminology & knowledge mismatches
  - data lifting & lowering

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

Data Level (OO) Mediation

- Related Aspects / Techniques:
  - Ontology Integration (Mapping, Merging, Alignment)
  - Data Lifting & Lowering
  - Transformation between Languages / Formalisms

- Terminology Mismatch Classification
  - Conceptualization Mismatches
    - same domain concepts, but different conceptualization
    - different levels of abstraction
    - different ontological structure
    - resolution only incl. human intervention
  - Explication Mismatches
    - mismatches between: T (Term used) D (definition of concepts) C (real world concept)
    - automated resolution partially possible
Ontology Mapping Language

- Language Neutral Mapping Language
  - mapping definitions on meta-layer (i.e. on generic ontological constructs)
  - independent of ontology specification language
  - “Grounding” to specific languages for execution (WSML, OWL, F-Logic)

- Main Features:
  - Mapping Document (sources, mappings, mediation service)
  - direction of mapping (uni- / bidirectional)
  - conditions / logical expressions for data type mismatch handling, restriction of mapping validity, and complex mapping definitions
  - mapping constructs:
    - classMapping, attributeMapping, relationMapping (between similar constructs)
    - classAttributeMapping, classRelationMapping, classInstanceMapping
    - instanceMapping (explicit ontology instance transformation)
  - mapping operators:
    - =, <, <=, >, >=, and, or, not
    - inverse, symmetric, transitive, reflexive
    - join, split

Mapping Language Example

![Ontology O1](Human
- name
Adult
Child)

![Ontology O2](Person
- name
- age
michael memberOf Person
- name = Michael Stollberg
- age = 28)

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

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

- requested and provided functionalities do not match precisely
- delta-relations = relation & difference of functional descriptions (semi-automatic)
- advanced techniques for Web service detection

<table>
<thead>
<tr>
<th>Delta-Situation (intersection)</th>
<th>Delta-Situation (formal)</th>
<th>Delta-Expression &amp; Relationships</th>
</tr>
</thead>
<tbody>
<tr>
<td>equal</td>
<td>$\phi = \psi$, i.e.</td>
<td>$\phi \setminus \psi$ = $\Delta$</td>
</tr>
<tr>
<td>subsume</td>
<td>$\phi \sqsubseteq \psi$, i.e.</td>
<td>$\phi \setminus \psi$ = $\Delta$</td>
</tr>
<tr>
<td>plugin</td>
<td>$\phi \sqsubseteq \psi$, i.e.</td>
<td>$\phi \setminus \psi$ = $\Delta$</td>
</tr>
<tr>
<td>intersection</td>
<td>$\exists x (\phi(x) \land \psi(x)) \land \neg(\forall x (\phi(x) \land \neg \psi(x))) = \phi(x) \setminus \psi(x)$</td>
<td>$\Delta = (\phi \sqsubseteq \psi) \lor (\neg \phi \sqsubseteq \psi)$</td>
</tr>
<tr>
<td>disjoint</td>
<td>$\phi \neq \psi$, i.e.</td>
<td>$\phi \setminus \psi = \Delta$</td>
</tr>
</tbody>
</table>

Functional Mediation Usage
Process Level Mediation

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

- partially resolvable by “process mediation patterns”

Addressed Process Mismatches
Process Mediation Example

Request

Processes Mediator

Service

- Itinerary [origin, destination, date]
- Time
- Price

- Origin
- Destination
- Itinerary [origin, destination]
- Date
- Itinerary [route, date, time, price]
Process Mediation Example

Summary

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

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

Standardization
Market Prospects
Future Issues

History I

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

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

• Rising attention in Research & Industry ..
History II

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

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

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

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

Standardization Efforts W3C

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

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

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

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

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

Industrial Efforts

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

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

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

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

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

Market Development (Gartner)

![Diagram showing the growth and development of B2B commerce](image-url)
## Market Volume

<table>
<thead>
<tr>
<th>MARKET</th>
<th>2006</th>
<th>2010</th>
<th>2015</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic Development</td>
<td>$50M</td>
<td>$0.4B</td>
<td>$2.0B</td>
</tr>
<tr>
<td>Semantic Infrastructure</td>
<td>$500M</td>
<td>$17.0B</td>
<td>$200.0B</td>
</tr>
<tr>
<td>Knowledge Work Automation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information-intensive</td>
<td>$1,100M</td>
<td>$30.0B</td>
<td>$250.0B</td>
</tr>
<tr>
<td>Knowledge-intensive</td>
<td>$350M</td>
<td>$4.5B</td>
<td>$40.0B</td>
</tr>
<tr>
<td>Systems That Know</td>
<td>$100M</td>
<td>$0.5B</td>
<td>$10.0B</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>$2,200M</td>
<td>$52.4B</td>
<td>$500.0B</td>
</tr>
</tbody>
</table>

Source: MILLS-OMIS

## Market Segmentation

$52.4 billion dollar market

### Horizontal

- Discover & Access: $2.7B
- Measuring: $4.1B
- Provision & Communication: $5.9B
- Integration & Interpretation: $6.6B
- New Infrastructure: $12.6B

**Total:** $52.4B

### Vertical

- Services: $4.5B
- Government: $11.1B
- Finance: $11.4B
- Trade: $9.8B
- Manufacturing: $9.6B

**Total:** $52.4B

### Regional

- Europe: $16.2B
- Asia: $15.2B
- North America: $15.2B
- ROW: $15.2B

**Total:** $52.4B

Source: RapraTech
Future Items

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

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

References & Acknowledgements
References

Foundations


Semantic Web Services

References

Semantic Web Services


References SWS: W3C Submissions

OWL-S

WSMO [see also www.wsmo.org]

SWSF

WSDL-S

References Discovery


References Discovery


References Composition


References Mediation


Acknowledgements

The WSMO working groups are funded by the European Commission under the projects DIP, Knowledge Web, SEKT, SWWS, and ASG; by Science Foundation Ireland under the DERI-Lion project; and by the Vienna city government under the FIT-IT Programme in the projects RW² and TCP.

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

Slide design by Roberta Hart-Hilber, DERI Austria