Semantic Web Services
Realisation der SOA Vision mit semantischen Technologien

Michael Stollberg
DERI Innsbruck – University of Innsbruck

Contents

• The Vision: semantics and services
• The Semantic Web
  – idea
  – ontologies
• Web Services
  – the SOA Vision
  – initial Web Service technology stack
  – deficiencies
• Semantic Web Services
  – idea & aims
  – WSMO
  – semantic SOA
• running examples from travel domain
The Vision

Static

WWW
URI, HTML, HTTP

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

Semantic Web
RDF, RDF(S), OWL
WWW
URI, HTML, HTTP

Semantic Web
RDF, RDF(S), OWL

Dynamic
Web Services
UDDI, WSDL, SOAP

enable computing over the Web
“Service Oriented Architectures” (SOA)

Static

The Vision

Automated Web Service Usage
„Semantic SOA“

The Vision
The Semantic Web

- next generation of the Internet (augmentation of the WWW)
- information has machine-processable and machine-understandable semantics
- ontologies as base technology for semantic interoperability
Ontology Definition

- unambiguous terminology definitions
- conceptual model of a domain (ontological theory)

formal, explicit specification of a shared conceptualization

- machine-readability with computational semantics
- commonly accepted understanding

Ontology Example

**Concept**
conceptual entity of the domain

**Property**
attribute describing a concept

**Relation**
relationship between concepts or properties

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

**Instance**
individual in the domain

```
holds(Professor, Lecture) =>
Lecture.topic = Professor.researchField
```

Ann memberOf student
name = Ann Lee
studentID = 12345
Ontology Languages

Requirements
- expressivity
- reasoning support
- web compliance

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

Ontology Technology

- **Ontology Reasoning**
  + advanced information processing
    - special requirements
      - large scale knowledge handling
      - fault-tolerant
      - stable & scalable inference machines

- **Ontology Management**
  - (collaborative) editing and browsing
  - storage and retrieval
  - versioning and evolution support
Ontology-Based Data Integration

- Ontology Integration Techniques

- integration on semantic level (domain independent)
- semi-automatic
  - human intervention needed for "integration decision"
  - graphical support for ontology mapping as central technique

Ontology Alignment
Ontology Mapping
Ontology Merging

Web Services
Web Services & SOA

- Web Service = program accessible over the Web
- Service-Oriented Architecture (SOA):
  dynamically find & invoke those Web services that allow to solve a particular request
- Web Service Technologies:
  1. **WSDL**  Web Service Description Language
     - in- and outgoing messages
     - technical access (port type, protocol, etc.)
  2. **SOAP**  XML data exchange protocol for the Web
  3. **UDDI**  registry for Web Services

The Promise of Web Services

web-based SOA as new system design paradigm

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

Running Example Setting

Objective: book trip for this workshop
- train or plane ticket
- hotel in Potsdam from Feb 19 – 21
- budget limitations
- preferences

SOA System
- train operators
- airlines
  - DB
  - ÖBB
  - HLX
  - BVG
  - German Wings
- hotel booking services
  - HRS
  - hotels.com
Aim: Realize the SOA Vision

- automate Web Service technologies by
  1. rich, formal annotation of Web Services
  2. inference-based techniques for automated discovery, composition, mediation, execution of Web Services

- integration with the Semantic Web
  - ontologies as data model
  - Web Services as integral part

- semantic SOA
  - also semantically describe client requests
  - automate complete SOA process
  - semantically enhance SOA technology
**Web Service Annotation**

a) Web Service Description Structure

- **Interface**
  - **Web Service Implementation**
    - (not of interest in Web Service Description)

b) Semantic Web Service Description Structure

- **Non-functional**
- **Functionality**

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

- **Aggregation**

**Semantic Web Service Technologies**

- **Request**
  - **Discoverer**
  - **Composer**
  - **Executor**
  - **Communication Conformance**
  - **Data Mediator**
  - **Process Mediator**

- **Service Repository**
  - Information lookup for particular service
  - Matchmaking R with all WS

- Uses:
  - if: composition needed
  - if: composition (executable)
  - if: directly usable
  - if: execution error
  - if: successful
  - if: compatible

- Submission:
  - else: try other WS

- Uses:
  - else: try other WS

- Uses:
  - else: try other WS

- Uses:
  - else: try other WS

- Uses:
  - else: try other WS

- Uses:
  - else: try other WS
• 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

www.wsmo.org
**WSMO Top Level Notions**

Objectives that a client wants to achieve by using Web Services

- Formally specified terminology 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*

---

**WSMO Web Service Description**

- **Capability**
  - functional description
  - realization of functionality by aggregation
  - functional decomposition
  - WS composition

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

- **Web Service Implementation**
  - Advertising of Web Service
  - Support for WS Discovery

- **Choreography** --- Service Interfaces --- Orchestration

WS - Advertising of Web Service
WS - Support for WS Discovery
WS - Orchestration
WS - Web Service Descrioption
DC + QoS + Version + financial
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: Capability

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

```prolog
capability VTAcapability
sharedVariables {?item, ?passenger, ?creditCard, ?initialBalance, ?reservationPrice}
precondition
definedBy
exists ?reservationRequest
(?reservationRequest[reservationItem hasValue ?item, passenger hasValue ?passenger, payment hasValue ?creditcard]
memberof tr#reservationRequest and
(?item memberOf tr#trip or ?item memberOf tr#ticket) and
?passenger memberOf pr#person and
?creditCard memberOf po#creditCard and
(?creditCard[type hasValue po#visa] or
?creditCard[type hasValue po#mastercard]) .
```
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.

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

Example: Capability

capability description (post-state)

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

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

- **Choreography** = how to interact with the service to consume its functionality
- **Orchestration** = how service functionality is achieved by aggregating other Web Services

**Diagram**

Choreography Interface

**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**

*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

=> WSMO solution: “ontologized Abstract State Machines”
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
choreography
stateSignature 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#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
endIf
```

WSMO Goals

<table>
<thead>
<tr>
<th>Client-Side</th>
<th>Service-Side</th>
</tr>
</thead>
</table>
| **Goal Template**
  generic objective description |
| **Goal Instance**
  concrete input |
| **Service detection** |
| **service usage** |
| **goal definition** |

**Client**
defines

**Domain Knowledge**

**Ontology**

**Service-Side**

**Goal Definition**

**Service detection**

**service usage**

**Web Service Implementation**

**functional**

**behavioral**

(not of interest here)
Goal Model

**Goal**
- domain: ontology
- objective: capability

**WG Mediator**
- source: goal
- target: webService
- usability: matchingdegree
- clientInterface: choreography

**Composite Goal**
- subGoals: goal workflow orchestration

**Goal Instance**
- template: goal
- input: inputbinding

**Desired Workflow**

**Basic Goal**

**Goal**
- buy train ticket in Germany
  - origin: o, destination: d
  - date-time: dt

**Goal Instance with Inputs**
- o = Munich, d = Berlin
- dt = 20070319-1030

**Client**

**WG Mediator**

**DB Ticketing**

**Client Interface**

**Design Time**

**Runtime**
Composite Goal

Flight-hotel booking with desired workflow

Goal

Flight Request

if hotel = Ø
flight.outwardArrival = hotel.arrival
flight information

Hotel Request

if flight = Ø
hotel information

Book Flight

Book Hotel

Web Service Discovery

detect directly usable Web services out of available ones

- Discovery Techniques (functional as primary focus)
  - 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
  - ...
Semantic Matchmaking

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


Web Service Composition

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

- composition of Web services is needed 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 to achieve goal

- composition techniques:
  
functional = composition wrt *functionalities*
  
behavioral = composition wrt *behavioral interfaces*

\[ \Rightarrow \text{need to be integrated:} \]
  
1. skeleton by functional composition
  
2. refinement + executable code by behavioral composition

**Procedure:**

- directly usable WS (discovery)?
  
  yes
  
  composition (functional)
  
  a) b) no
  
  composition skeleton
  
  no
  
  abort
  
  yes
  
  composition (behavioral)
  
  no
  
  abort
  
  executable composition
Choreography Discovery

determine behavioral compatibility

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

Goal Choreography Interface

<table>
<thead>
<tr>
<th>$\omega_0$</th>
<th>$\omega_1$</th>
<th>$\omega_2a$</th>
<th>$\omega_2b$</th>
<th>$\omega_3$</th>
<th>$\omega_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Omega_C^G(\omega_0) = {\varnothing}$</td>
<td>$\Omega_C^G(\omega_1) = {\text{request(out)}}$</td>
<td>$\Omega_C^G(\omega_2a) = {\text{offer(in)}, \text{changeReq(out)}}$</td>
<td>$\Omega_C^G(\omega_2b) = {\text{offer(in)}, \text{order(out)}}$</td>
<td>$\Omega_C^G(\omega_3) = {\text{offer(in)}, \text{conf(in)}}$</td>
<td></td>
</tr>
</tbody>
</table>

if $\varnothing$ then request
if $\text{cn1(offer)}$ then changeReq
if $\text{cn2(offer)}$ then order
if $\text{conf}$ then $\varnothing$

Start

$\omega_1(C)$

$\omega_2(C)$

$\omega_3(C)$

$\omega_4(C)$

Termination

valid choreography existent

WS Choreography Interface

<table>
<thead>
<tr>
<th>$\omega_0$</th>
<th>$\omega_1$</th>
<th>$\omega_2a$</th>
<th>$\omega_2b$</th>
<th>$\omega_3$</th>
<th>$\omega_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Omega_V^T(\omega_0) = {\varnothing}$</td>
<td>$\Omega_V^T(\omega_1) = {\text{request(in), offer(out)}}$</td>
<td>$\Omega_V^T(\omega_2a) = {\text{changeReq(in), offer(out)}}$</td>
<td>$\Omega_V^T(\omega_2b) = {\text{order(in), conf(out)}}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

if $\text{request}$ then offer
if $\text{order}$ then $\text{conf}$
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 resolve mismatches independent of content
- mediation cannot be fully automated (integration decision)

Levels of Mediation within Semantic Web Services:
1. Representation Level: heterogeneous Languages & Protocols
2. Data Level: heterogeneous Data Sources
3. Functional Level: heterogeneous Functionalities
4. Process Level: heterogeneous Communication Processes

Interoperability problems due to
- different representation formalisms
- different technical communication protocols

Adaptors for transformation
- syntactic transformation
- mappings between language constructs
- can be realized by ontology integration

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

- resolve semantic mismatches between terminologies
- realized by ontology integration
  - mappings between heterogeneous ontologies (design time)
  - data transformation (runtime)

### Ontology Mapping
- Mapping Rules

### Ontology Alignment
- Ontology A is made compatible to ontology B

### Ontology Merging

---

Mapping Language Example

**Ontology O1**
- Human
  - name
- Adult
- Child

**Ontology O2**
- Person
  - name
  - age
- michael
  - name = Michael Stollberg
  - age = 28

```xml
classMapping(unidirectional o2:Person o1.Adult
attributeValueCondition(o2.Person.age &ge; 18))
```

does 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**

- adjust goal description if it is not solvable
  1. weaken goal description
     * remove constraints so that G becomes solvable by W if not given a priori
  2. explicate usage conditions
     * additional constraints when W can solve G
- determined by delta-relations (semantic differences between requested and provided functionality)

**Process Level Mediation**

- not a priori compatible behavior interfaces for communication & information interchange => behavioral incompatibility
- partially resolvable by process mediation patterns
Patterns for Resolvable Mismatches

- Business Partner1
  - A
  - B
  - PM
- Business Partner2
  - B

- Business Partner1
  - A
  - B
  - PM
- Business Partner2
  - A

- Business Partner1
  - A and B
  - PM
- Business Partner2
  - A and B

- Business Partner1
  - A
  - B
  - AckA
- Business Partner2
  - A

Can resolve about 80% of process level mismatches

Unresolvable Process Mismatches

- Business Partner1
  - A
  - B
  - PM
- Business Partner2
  - B

- Business Partner1
  - A
  - B
  - PM
- Business Partner2
  - A

- Business Partner1
  - A
  - Ack
  - PM
- Business Partner2
  - ?
Process Mediation Example

REQUEST

itinerary [origin, destination, date]

REQUEST

itinerary [origin, destination, date]

REQUEST

itinerary [origin, destination, date]

REQUEST

itinerary [origin, destination, date]
Process Mediation Example

Processes Mediator

REQUEST

- itinerary [origin, destination, date]
- time
- price

SERVICE

- origin
- destination
- itinerary [origin, destination]
- date
- itinerary [route, date, time, price]
### Process Mediation Example

**Processes Mediator**

- **REQUEST**
  - origin
  - destination
  - itinerary [origin, destination, date]
  - time
  - price

- **SERVICE**
  - origin
  - destination
  - itinerary [origin, destination]
  - date
  - itinerary [route, date, time, price]

**OO Mediator**

**GG Mediator**

**WG Mediator**

### WSMO Mediators Overview

**OO Mediator**

**GG Mediator**

**WG Mediator**

**Legend**

- Technique used
- Imports / Reuses
- Correlation
The Web Service Execution Environment

WSMX (WSMO Reference Implementation)

Open source code base at SourceForge: [http://sourceforge.net/projects/wsmx/](http://sourceforge.net/projects/wsmx/)

Other WSMO Tools

[www.wsmo.org/tools](http://www.wsmo.org/tools)

- WSML (Specification Language)  [www.wsmo.org/wsml](http://www.wsmo.org/wsml)
  - conceptual language for WSMO
  - ontology language with several variants
- WSMO Editors:
  - WSML editors + validation
  - WSMO Studio
  - WSMO Visualizer
- Ontology Technology:
  - WSML Reasoner (for DL and LP)
  - Ontology Management Suite
  - Data Mediator (incl. Abstract Mapping Language)

all: Eclipse plugins & open source (LGPL licence)
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**
Other Approaches

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

---

**OWL and WSML**

- **OWL Full**  
  - full RDF(S) support

- **OWL DL**  
  - Description Logics

- **OWL Lite**  
  - subset of OWL DL

- **WSML DL**  
  - Description Logics

- **WSML Core**  
  - Logic Programming

- **WSML Full**  
  - First Order Logic

- **WSML Rule**

- **WSML Flight**

WSML aims at overcoming deficiencies of OWL
• Process Model for Web Services (FLOWS)
  • although self-contained, commonly understood as extension of OWL-S / refinement of Service Model

<table>
<thead>
<tr>
<th>Module</th>
<th>Explanation</th>
<th>Major Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOWS-Core</td>
<td>basic notions of services as activities composed of atomic activities</td>
<td>Service, Atom, Process</td>
</tr>
<tr>
<td>Control Constraints</td>
<td>common workflow-style process constructs, including OWL-S process model concepts.</td>
<td>Split, Sequence, Unordered Choice, Alternatives, Repetition</td>
</tr>
<tr>
<td>Ordering Constraints</td>
<td>allow specification of activities defined by sequencing properties of atomic processes</td>
<td>Orderability</td>
</tr>
<tr>
<td>Occurrence Constraints</td>
<td>support for non-deterministic activities within services</td>
<td>Occurrence Activity</td>
</tr>
<tr>
<td>Static Constraints</td>
<td>specify activities that are triggered by states of an overall system</td>
<td>Triggered Activity</td>
</tr>
<tr>
<td>Exception Constraints</td>
<td>basic infrastructure for modeling exceptions</td>
<td>Exception</td>
</tr>
</tbody>
</table>

**WSDL-S**

Semantic annotation of WSDL descriptions

1. annotate XML Schema with domain ontology

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

2. pre-conditions & effects for operations

```xml
<interface name="PurchaseOrder">
  <operation name="processPurchaseOrder" pattern="wsdl: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
<wsem: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

History & Standardization

Market Prospects
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>RDF</td>
<td>Resource Description Framework, revision</td>
<td>Recommendation</td>
<td>10 Feb 2004</td>
</tr>
<tr>
<td>OWL</td>
<td>ontology language for the Semantic Web</td>
<td>Recommendation</td>
<td>10 Feb 2004</td>
</tr>
<tr>
<td>SPARQL</td>
<td>RDF query language</td>
<td>Candidate Recommendation</td>
<td>06 April 2006</td>
</tr>
<tr>
<td>Web Services</td>
<td>SOAP 1.2</td>
<td>XML Messaging</td>
<td>24 June 2003</td>
</tr>
<tr>
<td></td>
<td>WSDL 2.0</td>
<td>Web service description</td>
<td>Candidate Recommendation</td>
</tr>
<tr>
<td></td>
<td>WS Addressing</td>
<td>endpoint &amp; message</td>
<td>Proposal Recommendation</td>
</tr>
<tr>
<td></td>
<td>WSDL-CDL</td>
<td>schema description</td>
<td>Candidate Recommendation</td>
</tr>
</tbody>
</table>

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

=> standardization need acknowledged, but no agreement yet on what & how

Web Services & SOA in Industry

- 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)
Estimated Market in 2010

$52.4 billion dollar market

Horizontal

Vertical

Regional

References
References Foundations


References Semantic Web Services


References SWS: W3C Submissions

OWL-S

WSMO [see also www.wsmo.org]

SWSF

WSDL-S

References Discovery


References Discovery


References Composition


References Mediation


References WSMO

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

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

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

The **WSMO working groups** are funded by the European Commission under the projects **DIP**, **Knowledge Web**, **SEKT**, **SWWS**, **ASG**, and **SUPER**; 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 **TSC**.

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