WSMO Training for DIP

DIP WP 14 Workshop
18-Jan-2005, Innsbruck

chairs: John Domingue, Liliana Cabral, Michael Stollberg
## DIP WP 14 Workshop Agenda

<table>
<thead>
<tr>
<th><strong>TUE, 18th Jan 05:</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>09 – 11</td>
<td>WSMO Tutorial</td>
</tr>
<tr>
<td>11 – 11.30</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>11 – 13</td>
<td>DIP &amp; WSMO Tools presentation &amp; demos</td>
</tr>
<tr>
<td>13 – 14.30</td>
<td>Lunch</td>
</tr>
<tr>
<td>14.30 – 15.30</td>
<td>DIP &amp; WSMO Tools presentation &amp; demos</td>
</tr>
<tr>
<td>15.30 – 15.45</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>15.45 – 18.00</td>
<td>Case Studies Sessions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>WED, 19th Jan 05:</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9.30 – 12</td>
<td>Case Studies Sessions (cont.)</td>
</tr>
<tr>
<td>12 – 12.30</td>
<td>Wrap Up &amp; Closing</td>
</tr>
</tbody>
</table>
WSMO Tutorial

Michael Stollberg, Titi Roman, Holger Lausen
Outline

1) WSMO Specification
   - WSMO 1.0
   - WSMO 1.1

2) News in WSMO
   - discovery
   - choreography
   - WSML

3) WSMO Use Case Walkthrough
   - WSMO D3.2 / D3.3
   - Use case definition / setup and resource modeling
WSMO Tutorial Part I

WSMO Specification

- WSMO 1.0
- WSMO 1.1

WSMO will be submitted to W3C as a members submission
WSMO Top Level Elements

Objectives that a client may have when consulting a Web Service

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

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

Connectors between components with mediation facilities for handling heterogeneities
WSMO 1.0

Non-Functional Properties

- Every WSMO element is described by properties that contain relevant, non-functional aspects of the item

- **Core Properties** (for every WSMO element)
  - Dublin Core Metadata Set
  - Version

- **Service Specific Properties:**
  - Quality of Service Information
  - Financial / contractual properties
nfp: Core Properties

ontology <"http://www.wsmo.org/ontologies/trainConnection">
nonFunctionalProperties
dc:title hasValue "International Train Connections Ontology"
dc:creator hasValue "DERI International"
dc:subject hasValues {"Train", "Itinerary", "Train Connection", "Ticket"}
dc:description hasValue "International Train Connections"
dc:publisher hasValue "DERI International"
dc:contributor hasValues ("Michael Stollberg",
  "http://homepage.uibk.ac.at/~C703225/foaf.rdf",
  "http://homepage.uibk.ac.at/~c703240/foaf.rdf",
  "http://homepage.uibk.ac.at/~c703262/foaf.rdf"}
dc:date hasValue "2004-10-08"
dc:type hasValue <"http://www.wsmo.org/2004/d2#ontologies">
dc:format hasValue "text/html"
dc:identifier hasValue <"http://www.wsmo.org/ontologies/trainConnection">
dc:source hasValue <"http://www.wsmo.org/2004/d3/d3.3/v0.1/20041119/resources/tc.wsml">
dc:language hasValue "en-US"
dc:relation hasValues {"http://www.daml.org/2001/06/itinerary/itinerary-ont",
  "http://daml.uml.edu/ontologies/italks/person",
  "http://www.wsmo.org/ontologies/datetime",
  "http://www.wsmo.org/ontologies/location",
  "http://www.daml.org/2001/02/geofile/geofile-ont",
  "http://www.wsmo.org/2004/d2#ontologies">
dc:coverage hasValue "ID:7029392 Name:World"
dc:rights hasValue <"http://www.deri.org/privacy.html">
version hasValue "$Revision: 1.6 $"
endNonFunctionalProperties
WSMO 1.0

nfp: Service Specific Properties

- Quality Aspects and other non-functional information of Web Services:
  - Accuracy
  - Availability
  - Financial
  - Network-related QoS
  - Performance
  - Reliability
  - Robustness
  - Scalability
  - Security
  - Transactional
  - Trust
WSMO 1.0

Ontologies

- Used as data model throughout WSMO
- Ontology elements: **concepts**, **relations**, **functions**, **axioms** and **instances**
- WSMO Ontology Design
  - **Modularization**: import / re-using ontologies, modular approach for ontology design
  - **De-Coupling**: heterogeneity handled by **OO Mediators**
- Ontology Specification Language: **WSML**
- Web Compatibility:
  - Namespaces
  - WWW Identification Concept (URI, Literal, Variable)
  - Basic Datatypes from XML Schema
WSMO 1.0

Goals

• **De-coupling of Request and Service**
  - Objective description independent of service usage
  - inherent support for discovery & service usage

• Constituting description elements:
  - **postcondition**: object of interest (computational aspects)
  - **effect**: conditions that have to hold after resolution (real world aspects)
  => *Only objective specification without regard to resolution by service*

• Usage:
  - Goal Ontologies (pre-existing Goal Templates)
  - Goal Resolution Process ‘open’ to implementations
WSMO 1.0

Web Services

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

Capability
- Advertising of Web Service
- Support for WS Discovery

Web Service Implementation
- Interaction Interface for consuming WS
  - Messages
  - External Visible Behavior
  - ‘Grounding’
- Functional description
  - Realization of WS by using other Web Services
    - Functional decomposition
    - WS Composition

Core + WS-specific

Choreography --- Interfaces --- Orchestration
WSMO 1.0

Web Service Capability

- **Non-Functional Properties**
- **Imported Ontologies**
- **Used Mediators**
  - *OO Mediator*: importing ontologies as terminology definition
  - *WG Mediator*: link to a Goal that is solved by the Web Service
- **Pre-Conditions**
  Input with conditions that web service expects in order to be able to provide its service (*computational aspects*)
- **Assumptions**
  Conditions that have to hold before the Web Service can be executed (*real world aspects*)
- **Post-Conditions**
  Result / Output of Web Service in relation to the input, and conditions on it (*computational aspects*)
- **Effects**
  Conditions / Changes on the state of the world that hold after execution of the Web Service (*real world aspects*)

**before execution**

**after execution**
WSMO 1.0

Choreography in WSMO

“Interface of Web Service for client-service interaction when consuming a Web Service”

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

• **Communication Structure**
  – messages sent and received
  – their order (messages are related to activities)

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

• **Formal Model**
  – allow operations / mediation on Choreographies
  – Formal Basis: Abstract State Machines (ASM)
WSMO 1.0

WSMO Orchestration

“Achieve Web Service Functionality by aggregation of other Web Services”

• **Orchestration Language**
  – decomposition of Web Service functionality
  – control structure for aggregation of Web Services

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

• **Proxy Technology**
  – Placeholders for used Web Services
  – Facility for applying the Choreography of used Web Services
WSMO 1.0

WSMO Orchestration Overview

Decomposition of the Web Service functionality into sub-functionalities

Proxies as placeholders for used Web Services

Control structure for aggregation of other Web Services
WSMO 1.0

Mediators

- For handling heterogeneity

Mediation Services
- as a Goal
- directly
- optionally incl. Mediation

Source Component
1 .. n

WSMO Mediator
uses a Mediation Service via

Target Component
1

Mediator Types: OO, GG, WG, WW
WSMO 1.0

Mediator Usage

Diagram showing the mediator usage in WSMO 1.0.
OBSO Mediator Example

Merging two ontologies

Goal:
“merge s1, s2 and s1.ticket subConceptOf s2.product”
WSMO 1.0

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”
mediate between a Web Service and Goal with a narrower desire

Web Service
“sell flight and train tickets”

 WG Mediator

Mediation Service

Goal
“buy a train ticket”

Goal:
“aTicket memberOf trainticket”

usesMediator

OO Mediator
(from above)

sources

Train Connection Ontology

Purchase Ontology

imports

imports

Train Ticket Purchase Ontology
WSMO 1.0

WSMO WW Mediators

• **Aim:**
  – Enable interoperability of heterogeneous Web Services
  ⇒ Support automated collaboration between Web Services
  – Related to Web Service Interfaces (not fully specified yet)

• **WW Mediators support all 3 Mediation Levels:**
  – *OO Mediators* for terminology import with data level mediation
  – Protocol Mediation for establishing valid multi-party collaborations
  – Process Mediation for making Business Processes interoperable
WSMO v1.1 – major issues

- Language / MOF layer Model for WSMO
  - Ontology Definition refined
  - Service Description refined
- Goal Definition changed
  - Logical language for formal statements
  - Non-functional properties refined

D2 v1.1, 24 December 2004, not final working draft
Language for defining WSMO

- Based of MOF (Meta Object Facility)
- 4-layer architecture for meta model specifications
  =>$\Rightarrow$ allows better understanding of subject of discourse

Language main constructs: **Class, Sub-Class, Attribute, type**
Goal Definition

• **WSMO 1.0 Goal definition rationale:**
  – goals should decouple requesters and providers
  – goals should state the desire / objective only, without regard to technical service usage issues

• **during testing and elaboration requests for changes / enhancement of notion of Goals arose:**
  – need for ‘goal input’ as counterpart for Capability matching
  – need for specifying preferences / constraints on accepted services
  – model for service usage process, identifying actors and actions (i.e. who is invoking & interacting with a Web Service)

=> update of Goal Notion
Goal Definition – Current Solution

Class goal
  hasNonFunctionalProperties type nonFunctionalProperties
  importsOntology type ontology
  usesMediator type {ooMediator, ggMediator}
  requestsCapability type capability multiplicity = single-valued
  requestsInterface type interface

• „requestsCapability“:
  – ’PAPE‘ for Goals as well
    • ’Goal Input‘ realized
    • easy mapping / connection to Capability
  – only 1 requested functionality possible for 1 Goal

• „requestsInterface“:
  – define supportable choreography (for conversation establishment)
  – express user preferences on service realization
WSMO Tutorial Part II

News in WSMO
discovery, choreography, WSML
News in WSMO

Discovery

Michael Stollberg

(based on WSMO D5.1, eds: Uwe Keller, Rubén Lara, Axel Polleres)
Discovery in WSMO

• Achievements:
  – elaborated framework for Web Service Discovery
  – prototype tests (Flora, theorem provers, DL Reasoner)
  – usability evaluation of existing tools / reasoners

• Content:
  – Web Service vs. Service
  – Discovery Process
  – Discovery Techniques
  – Matchmaking Notions
  – Approaches & Prototypes
Web Services and Services

• need to clarify terminology, we use the following:
  – **Service**
    a *provision of value in some domain* (not necessarily monetary, independent of how service provider and requestor interact)
  – **Web Service**
    *computational entity* accessible over the Internet using Web Service Standards & Protocols, *provides access to (concrete) services* for the clients.

=> Relation between the notions:
  – **Service** corresponds to a *concrete execution of a service* with given input values
  – **Web Service** provides a *set of services*; each invocation results in one service that is associated to the Web Service

*based on “Conceptual Architecture for Semantic Web Services”, C. Priest, ISWC 2004*
Overall Discovery Process

- **Requester Desire**
- **Goal Discovery**
- **Selected predefined Goal**
- **Goal refinement**
- **Web Service Discovery**
- **Concrete Capability (possibly dynamic)**
- **Still relevant WS**
- **Service Discovery**
- **Service to be returned**

**Predefined formal Goal**

**Available WS**

- **Ease of description**
- **Efficient Filtering**
- **Accuracy**

**Goal-Repos.**
Discovery Techniques

• Aim of Discovery: detect suitable (Web) Services to solve a Goal

• different techniques usable
  – Key Word Matching
    match natural language key words in resource descriptions
  – Controlled Vocabulary
    ontology-based key word matching
  – Logical Semantic Resource Descriptions
    … what WSMO aims at
Matchmaking Notions

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

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

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

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

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

\(\bigcirc = G\)
\(\bigcirc = WS\)
Approach I

Simple Semantic Descriptions

Information Space
all possible instances of used ontologies

Description Notion
all possible instances that satisfy restricted information space

Approach
- every single description notion denotes a precise ontology object with constraints
- meta-variable X denotes object (dynamically quantified by matchmaking notion)

Restrictions
- relations between description notions can not be defined explicitly
- causes repetitive object definitions

Prototype with VAMPIRE, SWF project

Goal Post-Condition

```
postcondition
definedBy
exists ?PurchaseItem(?PurchaseItem[
    item hasValue ?PurchaseFurniture
] memberOf swfmo:product) and
exists ?PurchaseFurniture(?PurchaseFurniture[
    material hasValues {wood},
] memberOf furn:chair) and
?X[
    purchaseItem hasValue ?PurchaseItem,
    buyer hasValue kb:MichaelStollberg,
    purchasePayment hasValue kb:MSCreditCard
] memberOf swfmo:purchaseContract .
```
Approach II

Complex Semantic Descriptions

• Approach:
  – Goal extent with input & output (PAPE define conditions on this)
  – Service description understood as state transition performed by execution (Pre, A = pre-state; Post, E = post-state)
  – Match if:
    • Goal Input can satisfy pre-state
    • if then the post-state can be achieved

• Realization and Prototype:
  – Transaction Logic
  – rule-based discovery
  – FLORA-2 prototype

• Problems:
  – Requires extension of WSMO conceptual model
  – complicated resource modeling
  – Several open issues

News in WSMO

Choreography

Titi Roman
WSMO Choreography – What to Model?

- VTA Example:

  When the service is requested
  Date
  Time
  Flight, Hotel
  Error
  Confirmation

  When the service requests
  Date, Time
  Hotel
  Error

VTA Service

- WSMO Choreography – model all visible interactions of the service (Orchestration shows how all the interaction are related)
Choreography in WSMO – Issues to Be Addressed

- **External Visible Behavior**
  - those aspects of the workflow of a Web Service where Interaction is required
  - described by basic process / workflow constructs: sequence, split, loop, parallel

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

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

- **Formal Model**
  - allow operations / mediation on Choreographies
  - Formal Basis: Abstract State Machines (ASM)
WSMO Choreography - Definition

- **State Signature** - defines the invariant elements of the state description.
- **State** - described by a set of instances
- **Guarded Transitions** - express changes of states by changing the set of instances

Class wsmoChoreography
  hasStateSignature type stateSignature
  hasState type state
  hasGuardedTransitions type guardedTransition
WSMO Choreography – State Signature

- **Non-Functional Properties** – same as WSMO nfps
- **Imported Ontologies** – same as WSMO Imported Ontologies
- **Used Mediators** – same as WSMO Used Mediators
- **Concepts in Choreography** - sub-Class of concepts defined in WSMO having their non functional properties extended with the attribute `mode`
- **Relations in Choreography** - sub-Class of relations defined in WSMO, having their non functional properties extended with the attribute `mode`
- **Functions in Choreography** - sub-Class of relations defined in WSMO, having their non functional properties extended with the attribute `mode`
- **Axioms** – same as WSMO Axioms
WSMO Choreography – the \textit{mode} Attribute

- The \textit{mode} attribute in the nfps of Concepts, Relations, and Functions:
  - \textbf{Static} - the extension of the concept, relation, or function cannot be changed
  - \textbf{Controlled} - the extension of the concept, relation, or function can only be changed by the service
  - \textbf{In} - the extension of the concept, relation, or function can only be changed by the environment + invocation mechanism
  - \textbf{Shared} - the extension of the concept, relation, or function can be changed by the service and the environment + invocation mechanism
  - \textbf{Out} - the extension of the concept, relation, or function can only be changed by the service + invocation mechanism
WSMO Choreography – State and Guarded Transitions

• State
  – described by a set of explicitly defined instances

• Guarded Transitions
  – used to express changes of states by means of rules, expressible in the following form:

\[
\text{if } \text{Cond} \text{ then } \text{Updates}.
\]

  • \text{Cond} - an arbitrary WSML axiom, formulated in the given signature of the state
  • \text{Updates} - consist of arbitrary WSMO Ontology instance statements
WSMO Orchestration

• Orchestration – how all the interaction of services are related

• Orchestration language
  – Decomposition of the Service functionality
  – Complex workflow constructs for aggregation of services: advanced branching and synchronization patterns, structural patterns, state-based patterns, etc
  – Complex methods for correlation, transactions, etc
WSMO Choreography – Further Plans

- Based on the conceptual model develop a user friendly language, which will include the intuitive workflow patterns (sequences, splits, loops, and maybe parallel)
- A graphical notation for the user language
- A formalization of the language for
  - Checking properties
  - Equivalence
  - Refinement, etc.
- Extension of this language to be suited for WSMO orchestration
WSMO Service Example (1)

A WSMO Ontology:

```xml
namespace <<http://www.wsmo.org/ontologies/tripReservationOntology#>>
...'
ontology <<http://www.wsmo.org/ontologies/tripReservationOntology#>>
  nonFunctionalProperties
  ...
  endNonFunctionalProperties

concept route
  nonFunctionalProperties
    dc:description hasValue "concept of a route between two stations"
  endNonFunctionalProperties
  sourceLocation type tc:station
  destinationLocation ofType tc:station

concept reservation
  nonFunctionalProperties
    dc:description hasValue "concept of reservation, containing a reservation holder"
  endNonFunctionalProperties
  reservationNumber ofType xsd:integer
  reservedRoute ofType route
  reservationHolder ofType prs:person
```

A WSMO Service:

```xml
namespace <<http://www.wsmo.org/ontologies/tripReservationService#>>
...'
service <<http://www.wsmo.org/ontologies/tripReservationService.wsml#>>
  nonFunctionalProperties
  ...
  endNonFunctionalProperties

  capability reservationServiceCapability

  interface reservationServiceInterface

  choreography reservationServiceChoreography
```
WSMO Service Example (2)

Service Capability:

capability reservationServiceCapability
    nonFunctionalProperties
        precondition
            definedBy
            (\text{Route memberOf is:route}
                \text{startLocation hasValue ?start,}
                \text{endLocation hasValue ?end})
            and
            (?start locatedIn = austria \text{ or }
            ?start locatedIn = germany)
            and
            (?end locatedIn = austria \text{ or }
            ?end locatedIn = germany)
            and
            (?connection (?start,?end))

    postcondition
        definedBy
        (\text{reservation memberOf is:reservation})
        and
        (\text{reservation route = ?route})
        and
        (\text{reservation number = ?route:connection})

    effect
        definedBy
        (\text{CreditCardInstance memberOf prs:CreditCard}
            \text{balance hasValue ?currentBalance})
        and
        (\text{reservationInstance memberOf is:reservation}
            \text{price hasValue ?reservationCost})
        and
        (?currentBalance.amount = ?currentBalance.amount - ?reservationCost.amount)

Service Choreography:

coreography reservationServiceChoreography
    stateSignature reservationServiceStateSignature
    guardedTransitions reservationServiceTransitionRules
WSMO Service Example (3)

State Signature:

```
stateSignature reservationServiceStateSignature
    nonFunctionalProperties
        ...
    endNonFunctionalProperties

hasConceptInChoreography route subClass ts:route
    nonFunctionalProperties
        ...
    endNonFunctionalProperties

hasConceptInChoreography creditcard subClass WSDL:check
    nonFunctionalProperties
        ...
    endNonFunctionalProperties

hasConceptInChoreography reservation subClass WSDL:reserveRoute
    nonFunctionalProperties
        ...
    endNonFunctionalProperties
```

Guarded Transitions (tentative):

```
guardedTransitions reservationServiceTransitionRules
    if routeInstance memberOf ts:route
        sourceLocation hasValue ?start,
        destinationLocation hasValue ?end) and
        (?start locatedIn = austria or ?start locatedIn = germany) and
        (?end locatedIn = austria or ?end locatedIn = germany) and
        (connection (?start ?end)) and
        (creditcardInstance memberOf creditcard)
        then
            (reservationInstance memberOf ts:reservation) and
            (reservationInstance reservedRoute = routeInstance) and
            (reservationInstance reservationNumber = createdNumber) and
            (reservationInstance reservationHolder = creditcardInstance.owner)
```

WSMO training for DIP
News in WSMO

WSML

Holger Lausen / Jos de Bruijn
WSML: Overview

• Introduction to WSML
• Rationale of WSML
• Syntaxes for WSML
• WSML Variants
  – WSML Core
  – WSML Flight
  – WSML OWL
  – WSML Full
• Conclusions
Web Service Modeling Language

- Four elements of WSMO:
  - Ontologies, Goals, Web Services, Mediators

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

- Provide a Web Service Modeling Language based on the WSMO conceptual model
  - Concrete syntax
  - Semantics

- Provide a Rule Language for the Semantic Web

- Many current Semantic Web languages have
  - undesirable computational properties
  - unintuitive conceptual modeling features
  - inappropriate language layering
    - RDFS/OWL
    - OWL Lite/DL/Full
    - OWL/SWRL
Syntaxes for WSML

- Human-readable syntax
  - Modular syntax
    * WSMO-syntax functions as “umbrella”
    * Modules for different WSML variants with clear layering
  - Syntax:
    * Inspired by OIL/OWL and F-Logic
    * Conceptual syntax
    * Logical Expression Syntax
  - Semantics is fixed in WSML variants

- XML syntax
  - Based on human-readable syntax

- RDF syntax
  - Based on human-readable syntax
WSML Conceptual Syntax for Ontologies

<table>
<thead>
<tr>
<th>Extra-Logical declarations</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Ontologies</td>
</tr>
<tr>
<td>• Namespaces</td>
</tr>
<tr>
<td>• Imported Ontologies</td>
</tr>
<tr>
<td>• Used Mediators</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-Functional Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Concepts</td>
</tr>
<tr>
<td>• Relations</td>
</tr>
<tr>
<td>• Instances</td>
</tr>
<tr>
<td>– Explicitly defined in ontology</td>
</tr>
<tr>
<td>– Retrieved from external instance store</td>
</tr>
<tr>
<td>• Axioms</td>
</tr>
</tbody>
</table>

Logical Declarations
WSML- Example

Conceptual Syntax

```
concept PlainTicket subConceptOf Ticket

nonFunctionalProperties
  dc:relation hasValue validates

endNonFunctionalProperties

origin ofType location

destination ofType location

departure ofType xsd#dateTime

arrival ofType xsd#dateTime
```

Logical Expression Syntax

```
axiom validDates

definedBy
  forall ?x,?y,?z(
    false impliedBy ?x[arrival hasValue ?y,
                      departure hasValue ?z] memberOf PlainTicket
                      and ?y < ?z) .
```
WSML Logical Expressions

- Frame and first order based concrete syntax
- Elements:
  - Function symbols (e.g. f())
  - Molecules (e.g. Human subClassOf Animal, John memberOf Human, John[name hasValue “John Smith”]).
  - Predicates (e.g. distance(?x,?y,?z))
  - Logical connectives (or, and, not, implies, equivalent, impliedBy, forall, exists)
- Example:

  ?x memberOf Human equivalent
  ?x memberOf Animal
  and ?x memberOf LegalAgent.
Variants of WSML

WSML-DL  \[\rightarrow\]  First-Order Logic  \[\rightarrow\]  WSML-Full

WSML-Core  \[\rightarrow\]  WSML-Flight  \[\rightarrow\]  WSML-Rule

Description Logics  \[\rightarrow\]  Logic Programming
**WSML-Core**

- Allows conceptual modeling of ontologies
- Based on OWL- (a subset of OWL, based on DLP fragment)
  - Efficient query answering
  - Allows to take advantage from optimization techniques developed in database research
  - Many existing implementations (e.g. XSB, OntoBroker, SWI-Prolog, KAON, DLV)
  - Import/export OWL ontologies
- Expressive enough for most current ontologies
- Can be used for limited goal/web service modeling
WSML-Flight

• Based on OWL Flight
  – Extends OWL Full (Datalog subset of OWL Full)
  – Adds UNA
  – Adds constraints
  – Adds non-monotonic features

• Is an extension of WSML Core
  – Adds limited support for nominals
  – Meta-modeling
  – Intuitive semantics for attributes
  – Extensive datatype support

• Language is based on Datalog with inequality, constraints and stratified negation
WSML-Rule

- Based on Logic Programming-variant of F-Logic and HiLog
- Minimal model semantics
- Implements default negation
- Allows unrestricted use of function symbols
- Full support for goal/web service modeling
WSML-DL

• WSML syntax – OWL semantics
• (to be developed)

• OWL epistemology:
  – Complete class definitions
  – Range/cardinality restrictions
WSML-Full

• Based on a combination of First-Order Logic and minimal model semantics and default negation
• Unifies rule language with first-order based language (e.g. OWL)
• Possible formalisms:
  – Auto-epistemic Logic
  – Default Logic
  – Circumscription
WSML Conclusions

• Formal languages for WSML
  http://www.wsmo.org/2004/d16/d16.1/v0.2
  – Defines Different Variants with Syntax and Semantics
    • (Core, Flight, Rule, DL, Full)

• Relation to other tools/formalisms
  http://www.wsmo.org/2004/d16/d16.2/v0.2/
  – WSML DL ~ OWL DL (e.g. Racer, FaCT, Pallet)
  – WSML Rule ~ F -Logic subset implemented in Flora2
  – WSML Flight ~ Datalog Engines
WSMO Tutorial Part III

WSMO Use Case Walkthru

Michael Stollberg
Aim & Outline

• Aim:
  – explain elaboration of WSMO use case
  – “How to get from a problem to a technical use case”
  => basis for private DIP Use Case Sessions

• Content:
  – WSMO Use Case Definition
  – Virtual Travel Agency Use Case (WSMO D3.3)
  – Lessons learned
General Structure of WSMO Use Cases

WSMO D3.2

• A WSMO / SWS Use Case should be:
  – situated in a specific application domain
  – address specific problems within SWS

• General Use Case Structure for easy understanding and comparability (based on W3C use case definition structure):
  1. Use Case Description: application field and problem identification
  2. Resource Overview: tabular definition of required WSMO elements
  3. Resource Models: complete resource models in WSML
  4. Technical Solution: techniques applied / developed for use case

=> Aim: define DIP Use Cases according to this template
   (to be discussed)
Use Case Description

WSMO D3.2

• Aim:
  – describe application area
  – identify problems to be addressed and solved by SWS technology

• Aspects:
  1. **Description:** overall scenario outline
  2. **Scope:** scope of application scenario and SWS specific issues to be addressed
  3. **Actors, Roles and Goals:** identification of
     - the actors involved in the scenario
     - their roles (i.e. what they do in the scenario)
     - their goals (i.e. what they want to achieve by participating in the scenario).
  4. **Usage Scenarios:** detailed description of envisioned system functionality, incl.:
     - activities / functional steps to be performed
     - SWS technological requirements
  5. **System Architecture:** outline general requirements and possible architecture of
     the respective SWS-based application.
VTA Use Case Description Example
WSMO D3.3
VTA Use Case Description Example

WSMO D3.3
WSMO VTA Use Case Elaboration

WSMO D3.3

• General:
  – served as a testing bed for definition / elaboration of WSMO aspects
  => took a very long time, and produced 13 version (!!) up to now
  – currently “pending”, although not completely finished
• What we learned (conc. ‘how to define use cases’):
  – Initial setup ‘easy’
  – Ontology design is crucial
    • everything is based on ontologies => need to be adequate
      *DF: “not crappy and proprietary data models, but good & agreed domain conceptualizations”*
    • modular ontologies, correct modeling very important
    • should be based / related to existing ontologies or conceptual domain models
  – Need for a stable syntax + resource description methodology
    • currently use WSML version 1.0, 20-09-2004 (Validator existing)
    • resource description methodology dependent on usage within tools,
      currently: set-based resource modeling as basis for ‘discovery with simple descriptions’
  – Several recursions necessary to make use case stable
Ontology Definition (what we did)

WSMO D3.3

- Ontology Creation Process:
  - Initial ontology (brainstorming session)
  - Study existing / related ontologies and conceptual models of domain
  - Resource description table + first stable version
  - Several recursions along with other resource definitions

- Example: Purchase Ontology
  - Aim / scope
    - Terminology definition of purchase within B2C settings
    - Should be generic, i.e. reusable as purchase ontology in different applications
    - Should be aligned with / based on existing ontologies or domain models
  - Main building blocks
    - Purchase, Purchase Order, Product
    - Payment
    - Delivery
  - Based on / related to RosettaNet PIP3A4 “PurchaseOrderRequest“
    - Existing standard for B2B purchase definition
    - Needed to be revised towards B2C scenario
  - Work group internal “defense & explanation” to reach a common understanding & agreement
  - … took a very long time until current status (which is not even considered final)
# Resource Overview PO Ontology

WSMO D3.3

<table>
<thead>
<tr>
<th>WSMO component type</th>
<th>ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Purchase Ontology</td>
</tr>
<tr>
<td>description</td>
<td>defines notions of purchasing, incl. buyer, seller, purchase order, purchase partners, payment notions, etc.; based on RosettaNet, but adopted for B2C setting</td>
</tr>
</tbody>
</table>
| imported ontologies / used mediators | - Date an Time Ontology  
- OWL Currency Mediator |
| main constructs           | main concepts: 
purchase, buyer, seller, contactInformation, purchaseOrder, product, paymentMethod, paymentTerms, delivery |
| WSML model                | Listing 3                     |
Goal Definition
WSMO D3.3

• Defining a Goal in natural language is quite easy, but to model it is quite tough
• Goal in VTA Use Case: “buy a train ticket for a trip from Innsbruck to Frankfurt July, 17th 2004, departure between 6 and 7 p.m."

• What needs to be defined:
  what shall be in the goal postcondition?
  ⇒ object that satisfies the desire and is expected as a computational result of the Web Service execution
  here: get a purchase document for a ticket for the train connection
  ... and in the goal effect?
  ⇒ object that satisfies the desire and is expected as an effect / change that holds in the real world after execution of the Web Service
  here: getting the purchased ticket delivered
Goal Definition - Namespaces

WSMO D3.3

namespace
   &lt;&lt;http://www.wsmo.org/2004/d3/d3.3/v0.1/20041119/resources/SpecificTrainTripInnsbruckFrankfurt#&gt;&gt;
dc:&lt;&lt;http://purl.org/dc/elements/1.1#&gt;&gt;
dt:&lt;&lt;http://www.wsmo.org/ontologies/dateTime#&gt;&gt;
tc:&lt;&lt;http://www.wsmo.org/ontologies/trainConnection#&gt;&gt;
po:&lt;&lt;http://www.wsmo.org/ontologies/purchase#&gt;&gt;
loc:&lt;&lt;http://www.wsmo.org/ontologies/location#&gt;&gt;
kb:&lt;&lt;http://www.wsmo.org/ontologies/kb#&gt;&gt;
wsml:&lt;&lt;http://www.wsmo.org/2004/d2/#&gt;&gt;
xsd:&lt;&lt;http://www.w3.org/2001/XMLSchema#&gt;&gt;

goal
   &lt;&lt;http://www.wsmo.org/2004/d3/d3.3/v0.1/20041119/resources/goal.wsml&gt;&gt;
Goal Definition - NFPs

WSMO D3.3

nonFunctionalProperties
dc:title hasValue "Buying a train ticket from Innsbruck to Frankfurt on..."
dc:creator hasValue "DERI International"
dc:subject hasValues {"Train Tickets", "Online Ticket Booking", "Train trip"}
dc:description hasValue "Express the goal of buying a concrete ticket for a train trip"
dc:publisher hasValue "DERI International"
dc:date hasValue "2004-10-04"
dc:type hasValue "http://www.wsmo.org/2004/d2#goals"
dc:format hasValue "text/html"
dc:identifier hasValue "http://www.wsmo.org/2004/d3/d3.3/v0.1/20041008/resources/goal.wsml"
dc:coverage hasValue "ID:7029392 Name:World"
dc:rights hasValue "http://deri.at/privacy.html"
version hasValue "$Revision: 1.4 $"
endNonFunctionalProperties
Goal Definition – Terminology Import

WSMO D3.3

importsOntologies
{
"http://www.wsmo.org/ontologies/trainConnection",
"http://www.wsmo.org/ontologies/purchase",
"http://www.wsmo.org/ontologies/location"
}

/*
no mediators used; the used OO Mediators are defined within
the ontologies
*/
postcondition axiom purchasingTicketForSpecificTrainTrip
nonFunctionalProperties
dc:description hasValue "The goal postcondition specifies that Tim Berners-Lee wants to
go buy a train ticket from Innsbruck to Frankfurt, departing from innsbruckHbf on 17th
July 2004 at 6 p.m."
endNonFunctionalProperties
definedBy
?TBLContactInformation{
?Purchase memberOf po:purchase[
po:purchaseorder hasValue ?Purchaseorder,
po:buyer hasValue ?Buyer ]
and ?Buyer memberOf po:buyer[
po:contactInformation hasValue ?TBLContactInformation,
po:billToAddress hasValue ?TBLAddress,
po:hasPayment hasValues {?PaymentMethod} ]
and ?TBLContactInformation memberOf po:contactInformation[
po:name hasValue "Tim Berners-Lee",
po:emailaddress hasValue "tbl@w3c.org" ]
and ?TBLAddress memberOf loc:address[
po:streetAddress hasValue "32 Vassar Street",
po:city hasValue boston,
po:state hasValue massachusetts,
po:country hasValue usa ]

Object of Desire

Buyer Infos
Goal Definition – Postcondition

WSMO D3.3

```sparql
and ?Purchaseorder memberOf po:purchaseOrder[  
  po:product hasValues {?Product},  
  po:payment hasValue ?PaymentMethod ]
and ?PaymentMethod memberOf po:creditCard[  
  po:type hasValue masterCard,  
  po:creditCardNumber hasValue 553546466867747,  
  po:holder hasValue "Tim Berners-Lee",  
  po:expMonth hasValue 09,  
  po:expYear hasValue 2007 ]
and ?Product memberOf po:product[  
  po:item hasValues {?Ticket} ]
and ?Ticket memberOf tc:ticket[  
  po:itinerary hasValue ?Itinerary ]
and ?Itinerary memberOf tc:itinerary[  
  po:passenger hasValue ?Passenger,  
  po:trip hasValue ?Trip ]
and ?Passenger memberOf tc:person[  
  po:name hasValue "Tim Berners-Lee" ]
and ?Trip memberOf tc:trainTrip[  
  po:start hasValue kb:innsbruckHBF,  
  po:end hasValue kb:frankfurtHBF,  
  po:departure hasValue ?Departure ]
and ?Departure memberOf dt:dateAndTime[  
  dt:date hasValue ?DepartureDate,  
  dt:time hasValue ?DepartureTime ]
and ?DepartureDate memberOf dt:date[  
  dt:dayOfMonth hasValue 17,  
  dt:monthOfYear hasValue 07,  
  dt:year hasValue 2004 ]
and ?DepartureTime memberOf dt:time[ dt:hourOfDay hasValue 18 ]).
```
Goal Definition – Effect
WSMO D3.3

```
effect axiom getTicketDelivered
  nonFunctionalProperties
    dc:description hasValue "The ticket is delivered via email."
  endNonFunctionalProperties
  definedBy
    exists ?Product, ?Buyer(
      ?Delivery memberOf po:onlineDelivery[
        po:deliveryItem hasValues {?Product},
        po:receiver hasValue ?Buyer,
        po:onlineDeliveryMethod hasValue "email" ]
    and ?Product memberOf po:product[
      po:item hasValues {?Ticket} ]
    and ?Ticket memberOf tc:ticket )
```
**Summary WSMO Modeling**

**WSMO D3.3**

- Currently **set-based resource modeling**: every description notion represents an object definition according to the domain ontologies that restricts the set of possible instances to those which satisfy the intended notion.

- **Problems:**
  - values / objects can not be transferred from one description notion to the other (addressed in WSMO D28)
  - Parsing needed from WSML to xxx that is processable by tools; WSML parser in progress (see WSML Validator)
Rest of VTA Use Case

WSMO D3.3

• VTA Web Service Capability (selling train tickets online)
  – Precondition: buyer information & itinerary wanted
  – Assumption: valid credit card (i.e. not expired)
  – Postcondition: purchase for a train ticket
  – Effect: delivery of train ticket

• “Goal Hierarchy”: 1 general Goal for purchasing tickets, and a concrete Goal that ‘instantiates’ the general goal (see before)

• different Mediators
  – define connections of resources only
  – No mappings / mediation services defined

• Related Ontologies