The Web Service Modeling Toolkit (WSMT)

Mick Kerrigan
Overview

- Towards Service Web
- WSMO, WSML and SEE
- The Web Service Modeling Toolkit
- Tools vs IDE
- Developing Semantic Descriptions in WSML
- Interacting with a SEE
- Creating Mediation Mappings between Ontologies
Web to Web 2.0

Web

UGC

Web 2.0

Linkedin

flickr

YouTube

Technorati

delicio.us

social bookmarking

Asian Autumn School on the Semantic Web 2007
Wednesday, 7th November
2007, 14:30 - 16:00

making semantics real.
Asian Autumn School on the Semantic Web 2007
Wednesday, 7th November
2007, 14:30 - 16:00

making semantics real.
The Semantic Web

Asian Autumn School on the Semantic Web 2007
Wednesday, 7th November 2007, 14:30 - 16:00
Semantic Web Services

Services

Web

Dynamic

Semantics

UGC

Web 2.0

Web 3.0

Semantic Web Services

Semantic Web

Web of Applications
Service Web

Semantic Web

Web

Service Web

Web 2.0

Web 3.0

Dynamic

UGC

Semantics

Services

Web of Applications

Asian Autumn School on the Semantic Web 2007
Wednesday, 7th November 2007, 14:30 - 16:00
making semantics real.
WSMO Recap

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
WSMO, WSML and SEE

Conceptual Model for SWS

Formal Language for WSMO

Ontology & Rule Language for the Semantic Web

Execution Environment for SWS
• A language framework for representing the elements of the WSMO conceptual model

• Language variants covering:
  – Description Logics
  – Logic Programming
  – First-Order Logic

• An expressive rule language for the Semantic Web

• An attempt to combine description logic and logic programming in one logical framework
WSML Layering
Human readable Syntax

wsml-variant "http://www.wsno.org/wsml/wsml-syntax/wsml-flight"
namespace {
   _ "http://www.simpsons.org/ontologies/",
   dc _ "http://purl.org/dc/elements/1.1/"
}

ontology simpsons
   nonFunctionalProperties
      dc#creator hasValue "Mick Kerrigan"
   endNonFunctionalProperties

concept actor
   hasName ofType _ string

concept character
   hasName ofType _ string
   hasActoer ofType actor

instance dan_castellanata memberOf actor
   hasName hasValue "Dan Castellaneta"

instance homer_simpson memberOf character
   hasName hasValue "Homer Simpson"
   hasActor hasValue dan_castellanata
wsml-variant _“http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"
namespace { _“http://www.amazon.com/”,
    dc _“http://purl.org/dc/elements/1.1”}

webservice amazonWebService

capability amazonCapability
    precondition amazonPrecondition
        definedBy ...
    postcondition amazonPostcondition
        definedBy ...
    assumption amazonEffect
        definedBy ...
    effect amazonEffect
        definedBy ...

interface amazonInterface
    choreography amazonChoreography
        ...
    orchestration amazonOrchestration
        ...
• WSML2Reasoner framework provides access to underlying reasoners for the different WSML Variants
  – Description Logics – Pellet
  – Logic Programming – IRIS, MINS, KAON2
  – First-Order Logic – SPASS + T

• Provides validation, normalization and transformation functionalities needed to transform from the WSML Syntax of a given variant to the syntax expected by the underlying reasoner
Semantic Execution Environments

- Core services within a Semantically Enabled Service-orientated Architecture (SESA)

- Enable the automation of previously human intensive tasks when building applications with a Service Oriented Architecture
  - Discovery: Determine usable services for a request
  - Composition: Combine services to achieve a goal
  - Ranking and Selection: Choose appropriate service for the job
  - Mediation: Solve mismatches to enable interoperability
  - Invocation: Execute entry points on the service
- Find Semantic Web Services that can totally or partial fulfill the end users Goal
Discovery

- Exact Match
- Plugin Match
- Subsumption Match
- Intersection Match
- No Match
• Combine a number of Semantic Web Services together to fulfil the end users Goal
• Combine a number of Semantic Web Services together to fulfil the end users Goal
• Choosing the most appropriate Web Service that meets the end users non-functional requirements
• Data Mediation for resolving terminological mismatches and enabling interoperability at the data level
  – Ontology Merging
  – Ontology Alignment
  – Instance Transformation
• Process Mediation for resolving communication mismatches, establishing behavioural compatibility and allowing interoperability at the process level
Invocation

• Execution of selected Web Services’ Choreography or Orchestration

• Multiple entry points of multiple concrete Web services may be invoked involving:
  – Lowering Ontological instance data to XML Messages
  – Lifting resulting XML Messages back to Ontology Instances
Semantic Web Service Life Cycle

- Concept Exploration
- Requirements
- Design
- Implementation
- Test
- Installation and Checkout
- Operation and Maintenance
- Retirement
The Web Service Modeling Toolkit (WSMT)

• The WSMT is an Integrated Development Environment (IDE) for the development of Semantic Web Services
• Aims to support the developer through the Software Development Cycle (SDC) of Semantic Web Services
  – Improve Developer Productivity
  – Aid in adoption of WSMO, WSML, SEE
  – High quality tools
  – Eclipse based
Why Tools?

• First tools included Unix command line tools that could be combined together with pipes
  – grep, awk, make

• Tool support reduces length of tasks
  – Long involved tasks can be reduced to seconds
  – Developer boredom reduced

• Visual and Non Visual Tools needed
  – Non Visual: Compilers, validators, debuggers
  – Visual: Editors, Browsers, Feedback, Testers
Why IDE?

- IDE’s seamlessly integrates individual tools
  - Gives a face to textual tools, hiding their complexity
  - Enables interoperation between previously separate tools
  - Reduces training costs (Increased ROI)
  - Removes switching back and forth between applications

- Tool is to IDE as
  - HTML Validator is to Dreamweaver
  - Java Compiler is to Eclipse JDT
  - WSML2Reasoner Framework is to WSMT
WSMT Functionality

- Development of WSMO Semantic Descriptions through WSML
  - Ontologies
  - Goals
  - Web Services
  - Mediators

- Interfacing with Semantic Execution Environments
  - WSMX
  - IRSIII

- Creation of Mediation Mappings between Ontologies
  - Abstract Mapping Language (AML)
• Semantic Execution Environments need Ontologies, Goals, Web Services, and Mediators in order to function

• Provide support to the developer in creating these descriptions

• Provide mechanisms for browsing semantic descriptions to aid in developer understanding

• Abstract the developer from the underlying syntax

• Assist in the validation and testing of semantic descriptions
• Abstracting from syntax is good but…
• Existing developers familiar with the syntax
• Certain tasks are just easier with a textual representation
• WSML Human Readable Syntax is designed to be light

Must support the more experienced developer
WSML Text Editor

Syntax Highlighting
Syntax and Content Autocompletion
Error Notification
Content Folding
Bracket Matching
WSML Form based Editor

- Abstracts developers from the WSML syntax allowing them to focus on the modeling task at hand
  - Improved Developer focus
  - Reduced Errors in semantic descriptions
  - Less keystrokes improves speed of creation
- Descriptions are broken up into tabs to keep the forms small
- Forms consist of Text fields, combo boxes and tables
WSML Form based Editor

WSML Form Based Editor

http://example.org/uaa

Capability

Name: http://example.org/uaacap
Shared Variables: x

Postconditions

Assumptions

Effects

Postcondition Expressions

memberOf _"http://example.org/trip" and forall ?from
    ( (?["http://example.org/from" hasValue "from"] implies
        ?frommemberOf _"http://example.org/PlaceInEurope" )
    and forall ?to
        ( (?["http://example.org/to" hasValue "to"] implies
            ?tomemberOf _"http://example.org/PlaceInEurope" )
    and forall ?veh
        ( (?["http://example.org/vehicle" hasValue ?veh] implies
            ?vehmemberOf _"http://example.org/Car" )

Header | Webservice | Capability
• In Textual, Form or Tree based representations it is hard to see the full relationship between entities
• Graph based representations give a better “Feel” for the complexities of a semantic description
• However normally visualizers are bolted on top of existing tools
• The WSML Visualizer provides editing and browsing support in one tool
• Immediate feedback to the developer as semantic descriptions are being created
WSML Visualizer

Graph Manipulation
Full Editing Support
Filtering
Instance Clustering
Semantic Levels
Semantic Highlighting
Outline View

- Eclipse views enhance the functionality of editors for different file formats
- The outline view gives a structured view of a WSML file
- Can be used in conjunction with any of the editors in the WSMT
- Bidirectional updates ensures that the selection in each editor and the view is up to date at all times
- Provides a browsing mechanism for any WSML description
Validation

- WSMO4J parser used to validate syntax
- WSMO4J validator used to validate semantics
  - Ensures features within the semantic description match that of the specified WSML Variant (Errors)
  - Checks for unrecommended usage of WSML Features (Warnings)

- All files automatically checked as they are changed
- Immediate feedback to the user in each editor
- Additional mechanisms for seeing errors
  - Problems view
  - WSML Navigator
Validation & the Problem View

<table>
<thead>
<tr>
<th>Description</th>
<th>Resource</th>
<th>In Folder</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axiom 'principles_work_too' - Inadmissible Atomic formula</td>
<td>the-simpsons-ontology.wsmi</td>
<td>Examples/The Simpsons</td>
<td>Line 21</td>
</tr>
<tr>
<td>Axiom 'souses_are_in_love' - Inadmissible Atomic formula</td>
<td>the-simpsons-ontology.wsmi</td>
<td>Examples/The Simpsons</td>
<td>Line 27</td>
</tr>
<tr>
<td>Ontology 'AboutTrips' - concept Trio2 not explicitly declared!</td>
<td>Trips.wsmi</td>
<td>Examples/Lightweight DL...</td>
<td>Line 4</td>
</tr>
</tbody>
</table>
WSML Navigator

Asian Autumn School on the Semantic Web 2007
Wednesday, 7th November 2007, 14:30 - 16:00
• Testing software usually involves deploying it and ensuring that it functions as expected
• Involves a costly Deploy-Test-Redeploy cycle
• Support within an IDE for testing software in its natural habitat can vastly reduce this iterative process
  – Reduces the cost of development
  – Improves developer productivity
  – Reduced developers involvement in tedious tasks
• Correctness of a semantic description is more than just having a valid description
Testing Ontologies

• Ontologies underlie every other semantic description in WSML
• The developer needs to be sure that each ontology behaves as expected when used in a reasoner
  – Is the ontology consistent?
  – Does is answer queries in the manner expected?
• Access to reasoners for each of the WSML Variants is thus required within the WSMT
• All users to perform reasoning operations over the ontology currently being edited
Ontology selection
Reasoner selection
Syntax Highlighting
Interfacing with editors
Testing Web Services and Goals

- A Semantic Web Service that does not match the Goals it is expected to match could result in the loss of a lot of money
- Developers need to ensure that the Web Service descriptions that create match Goals as expected
- Tool support reduces the number of interactions with a testing SEE
- Quite likely that provider will issue sample Goals with their Web Service descriptions.
- Ensuring your Web Service descriptions are found by your competitors sample Goals could provide a competitive advantage.
WSML Discovery View

Goal Selection

Web Service Selection

Discovery Selection

Type of Match

Interfacing with editors
In order for a SEE to correctly function the necessary Ontologies, Goals, Web Services and Mediators need to be available to it.
Manually deploying descriptions to a SEE or manually retrieving them in order perform maintenance is a tiresome and lengthy process.
Automated tools for interfacing with the Web Services exposed by a SEE enable these actions to be reduced to one or two clicks of a mouse.
The SEE perspective contains all the functionality necessary to deliver this tool support to the developer.
Browsing WSML in a SEE

Asian Autumn School on the Semantic Web 2007
Wednesday, 7th November 2007, 14:30 - 16:00
Retrieving WSML from a SEE
Storing WSML to a SEE
Invoking a SEE

Invoking a SEE
Ontology Mediation

• Service Consumers and Providers may not agree on terminology

• Instance transformation can transform instances from the consumers ontologies to instances of the providers ontologies

• Automatic approaches use algorithms to detect alignment between the source and target ontologies
  – Low precision
  – No developer involvement required

• Manual approaches rely on the developer creating the alignment by hand
  – Can get 100% accuracy
  – A lot of work needed to create all the mappings
Ontology Mediation (WSMX/WSMT)

Source Ontology

Target Ontology

Source Instance

Target Instance

Data Mediation

Mappings

Design-time Component

Run-time Component

Storage

Ontology Mediation (WSMX/WSMT)
Abstract Mapping Language

• Language Neutral Mapping Language
  – mapping definitions on meta-level
  – independent of ontology specification language

• Grounding can later be done to specific language for execution
  – WSML
  – OWL
  – F-Logic
This allows to transform the instance ‘adrian’ of concept person in ontology O2 into a valid instance of concept ‘adult’ in ontology O1.
An editor with for creating mappings using drag and drop

Different views all for different types of mappings to be created:
- Part of view: C2C, A2A, C2A, A2C
- Instance of view: conditional mappings
- RelatedBy view: R2R

Guides the developer through the process of creating these mappings using embedded suggestion algorithms
View based Editor (PartOf View)
View based Editor (InstanceOf View)
Mapping Views

- As ontologies become bigger mappings can be harder to see

- View based editor also obscures the type of the mapping

- Provide the developer with a mechanism for quickly seeing mappings by type

- Provide a mechanism for deleting one or more mappings
### Mapping Views (Concept2Concept)

<table>
<thead>
<tr>
<th>Source Concept</th>
<th>Target Concept</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Travelloucher</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>currency</td>
<td>boolean</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>string</td>
<td>string</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>integer</td>
<td>integer</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>terms</td>
<td>payment</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>date</td>
<td>date</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>basket</td>
<td>name</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>cost</td>
<td>payment</td>
<td>No conditions associated</td>
</tr>
</tbody>
</table>

**Mapping Views (Concept2Concept)**

<table>
<thead>
<tr>
<th>Source Concepts</th>
<th>Target Concepts</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Traveloucher</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>Currency</td>
<td>Boolean</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>String</td>
<td>String</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>Integer</td>
<td>Integer</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>Terms</td>
<td>Payment</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>Date</td>
<td>Date</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>Basket</td>
<td>Name</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>Cost</td>
<td>Payment</td>
<td>No conditions associated</td>
</tr>
</tbody>
</table>
### Mapping Views (Attribute2Attribute)

<table>
<thead>
<tr>
<th>Source Attributes</th>
<th>Target Attributes</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ticket] arrival_time =&gt; time]</td>
<td>[traveler] arrival_time =&gt; time]</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>[ticket] departure_date =&gt; date]</td>
<td>[traveler] departure_date =&gt; date]</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>[cost] amount =&gt; integer]</td>
<td>[payment] amount =&gt; integer]</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>[ticket] type =&gt; string]</td>
<td>[traveler] type =&gt; string]</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>[date] day =&gt; integer]</td>
<td>[date] day =&gt; integer]</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>[cost] hasCurrency =&gt; currency]</td>
<td>[payment] hasCurrency =&gt; boolean]</td>
<td>ValueCondition(== true on ((payment) in [currency])</td>
</tr>
<tr>
<td></td>
<td>ValueCondition(== true on ((cost) hasCurrency =&gt; currency))</td>
<td></td>
</tr>
<tr>
<td>[ticket] HasName =&gt; string]</td>
<td>[name] firstName =&gt; string]</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>[date] month =&gt; integer]</td>
<td>[date] month =&gt; integer]</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>[time] minutes =&gt; time]</td>
<td>[time] minutes =&gt; integer]</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>[cost] hasCurrency =&gt; currency]</td>
<td>[payment] hasCurrency =&gt; boolean]</td>
<td>ValueCondition(== true on ((payment) in [currency])</td>
</tr>
<tr>
<td></td>
<td>ValueCondition(== true on ((cost) hasCurrency =&gt; currency))</td>
<td></td>
</tr>
<tr>
<td>[ticket] issuing_terms =&gt; terms]</td>
<td>[traveler] terms =&gt; string]</td>
<td>No conditions associated</td>
</tr>
</tbody>
</table>

Making Semantics Real.
• Developers need to be confident in the mappings they create

• Testing involves ensuring that a given set of source instances translate into the expected set of target instances

• Very time consuming task involving a lot of tedious work

• Automation of comparison enables engineer to quickly perform tests
  – Ensure mappings still valid as ontologies evolve
  – Ensure mappings behave as expected on different reasoners
// INPUT

```xml
<instance my_ticket_1 memberOf travel1#ticket
   travel1#type hasValue "flight"
/>
```

// EXPECTED OUTPUT

```xml
<instance expected_travelVoucher_1 memberOf travel2#travelVoucher
   travel2#type hasValue "flight"
/>
```

// INPUT

```xml
<instance my_ticket_2 memberOf travel1#ticket
   travel1#type hasValue "flight"
   travel1#firstName hasValue "Adrian"
   travel1#lastName hasValue "Nocan"
/>
```

// OUTPUT

```xml
<instance expected_travelVoucher_2 memberOf travel2#travelVoucher
   travel2#source hasValue expected_voucher_2
   travel2#type hasValue "flight"
>
<instance expected_name_2 memberOf travel2#name
   travel2#firstName hasValue "Adrian"
   travel2#lastName hasValue "Nocan"
>
… and then let’s use the toolkit!