DERI INNSBRUCK

Leopold Franzens
Universität Innsbruck

The Web Service Modeling Toolkit (WSMT)

Mick Kerrigan

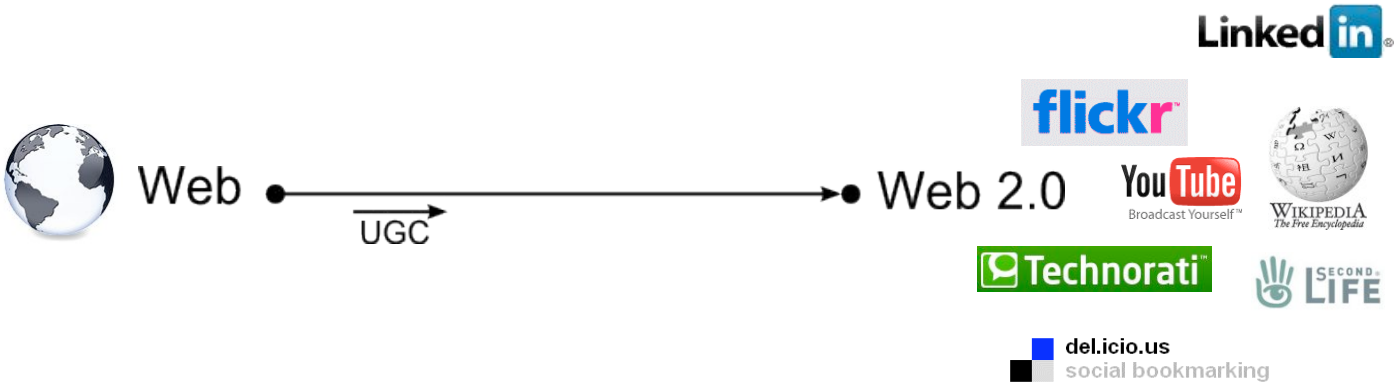
Asian Autumn School on the Semantic Web 2007
Wednesday, 7th November 2007,
14:30 - 16:00

making semantics **real.**

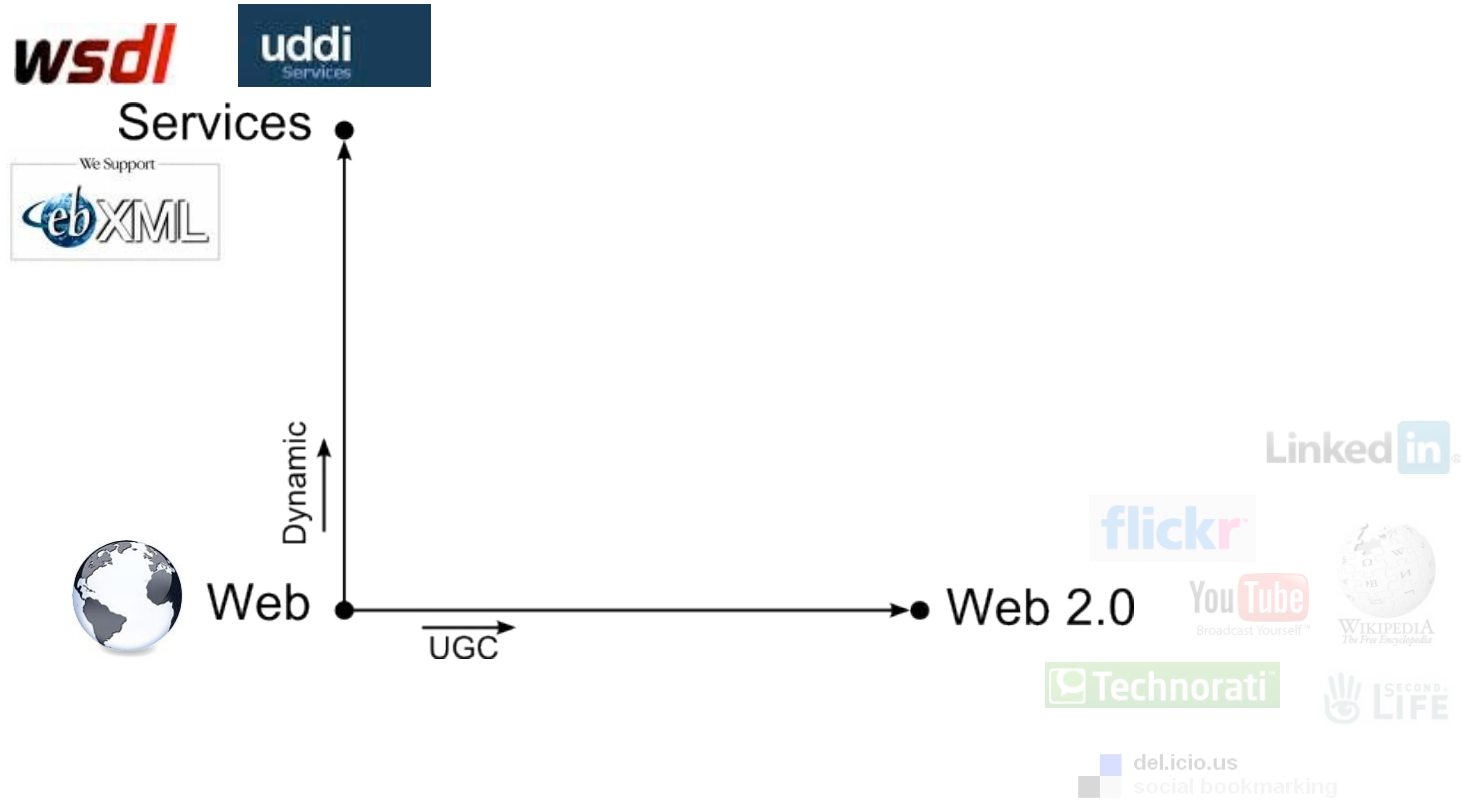
Overview

- 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

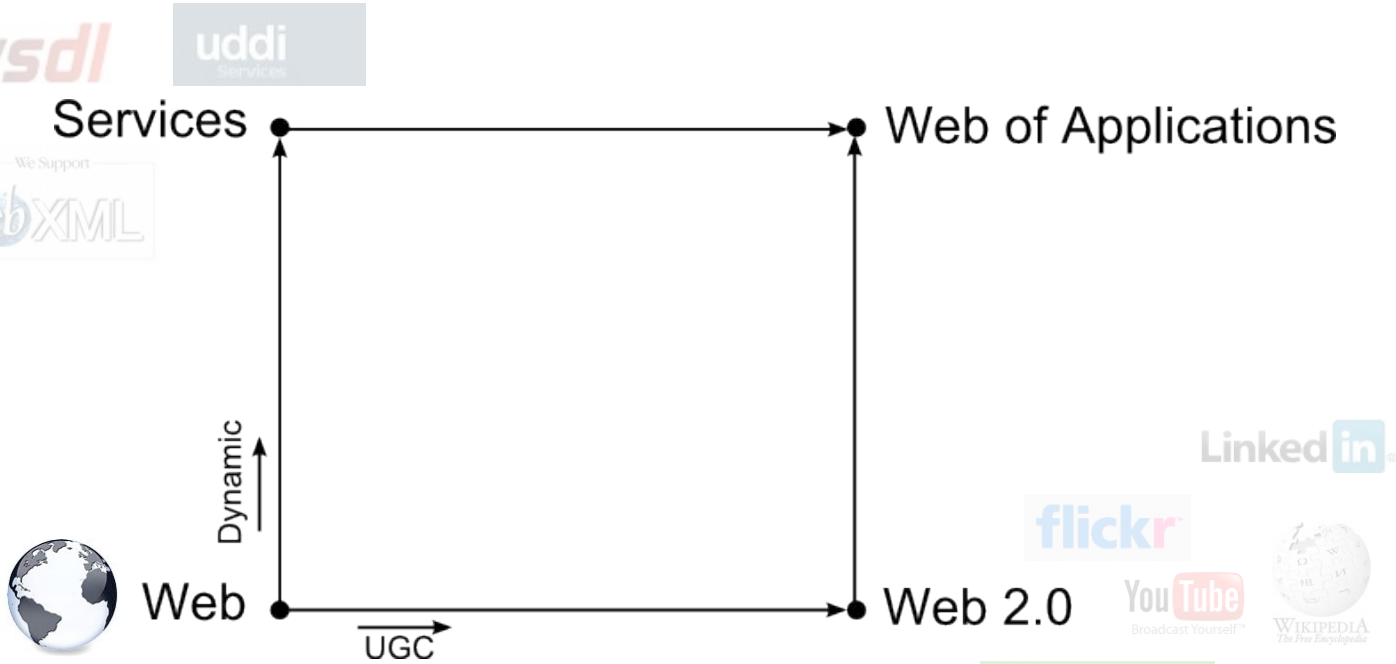
Web to Web 2.0


Web
UGC
Web 2.0
LinkedIn
flickr
YouTube
Broadcast Yourself
Wikipedia
The Free Encyclopedia
Technorati
Second Life
del.icio.us
social bookmarking

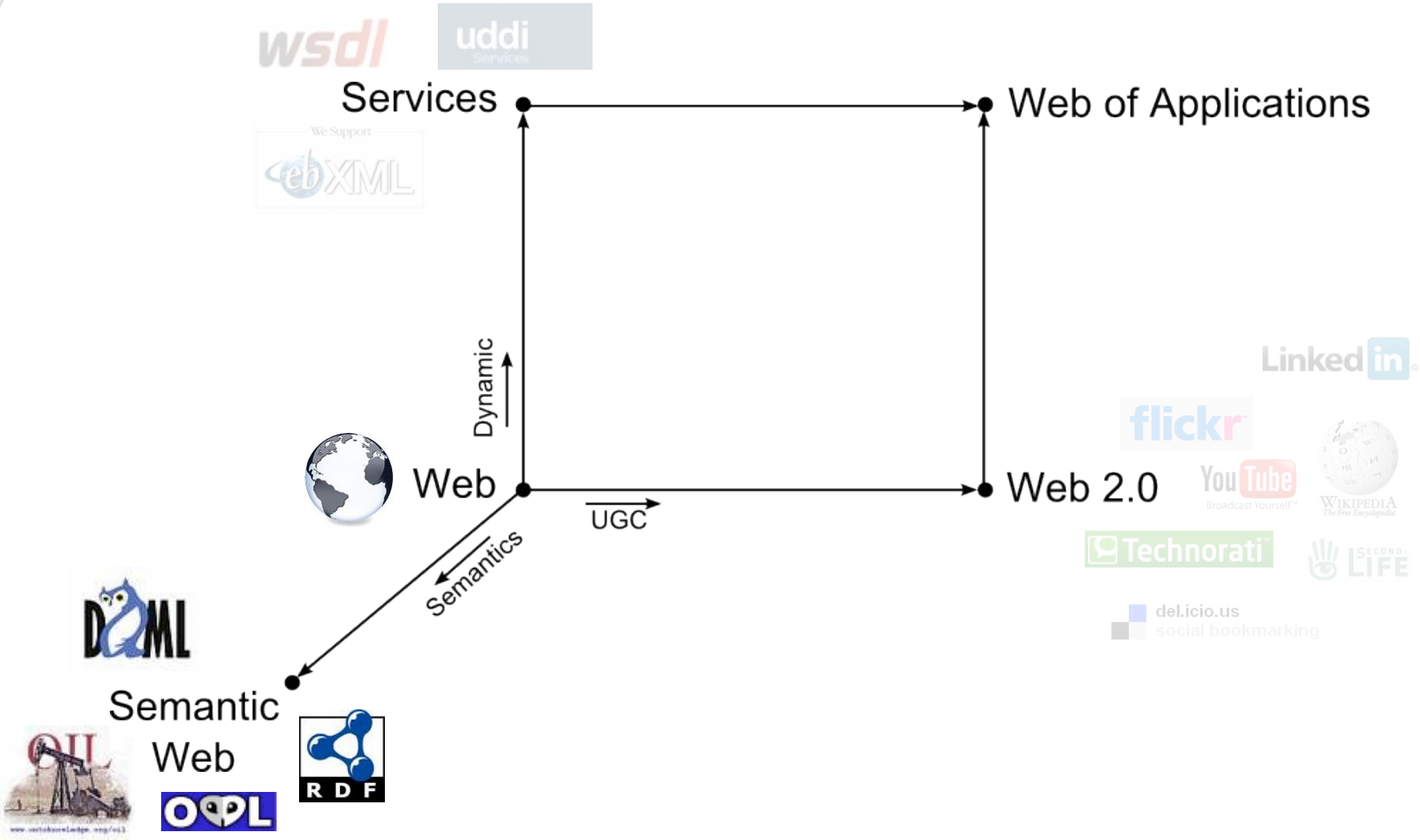
Web Services



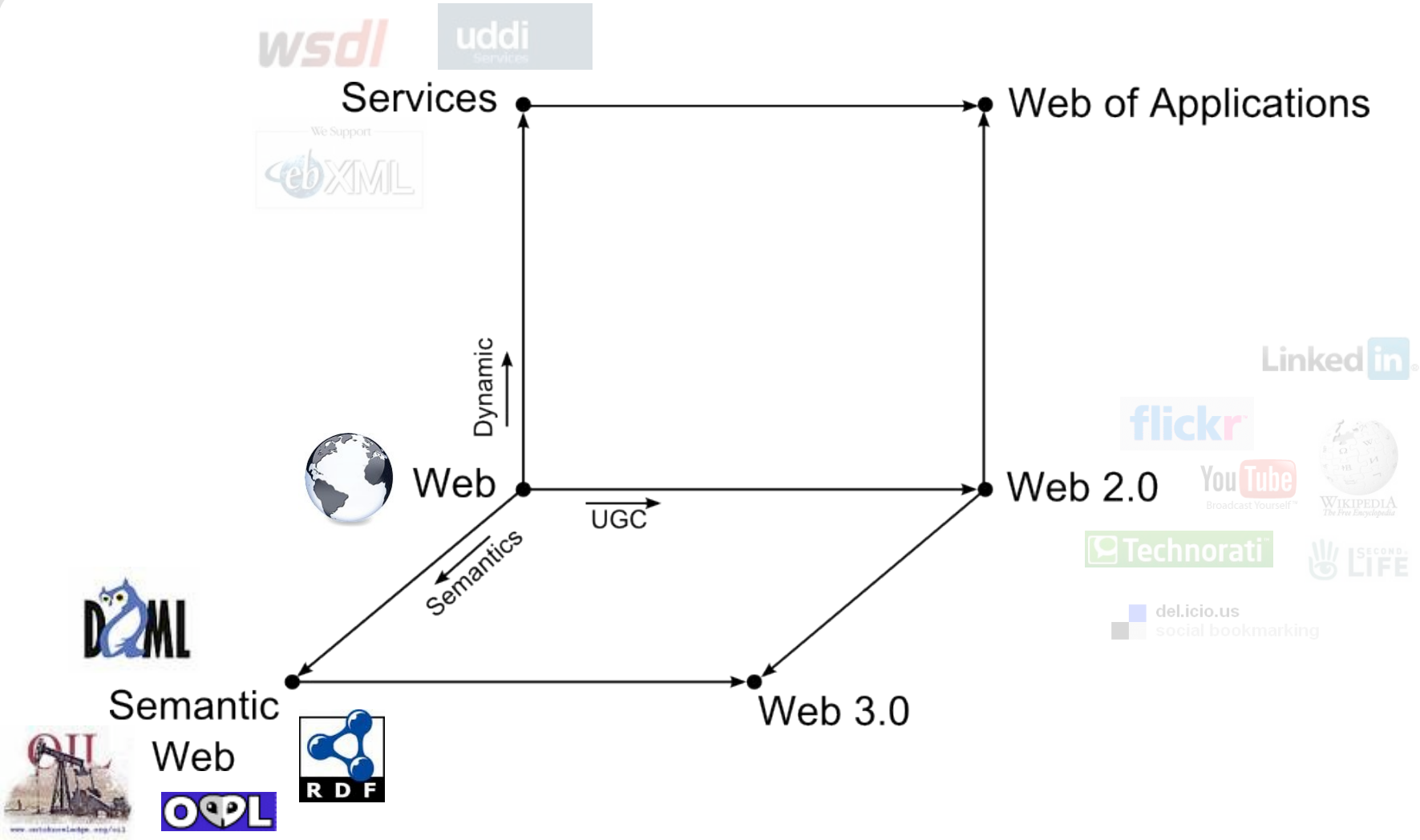
A Web of Applications



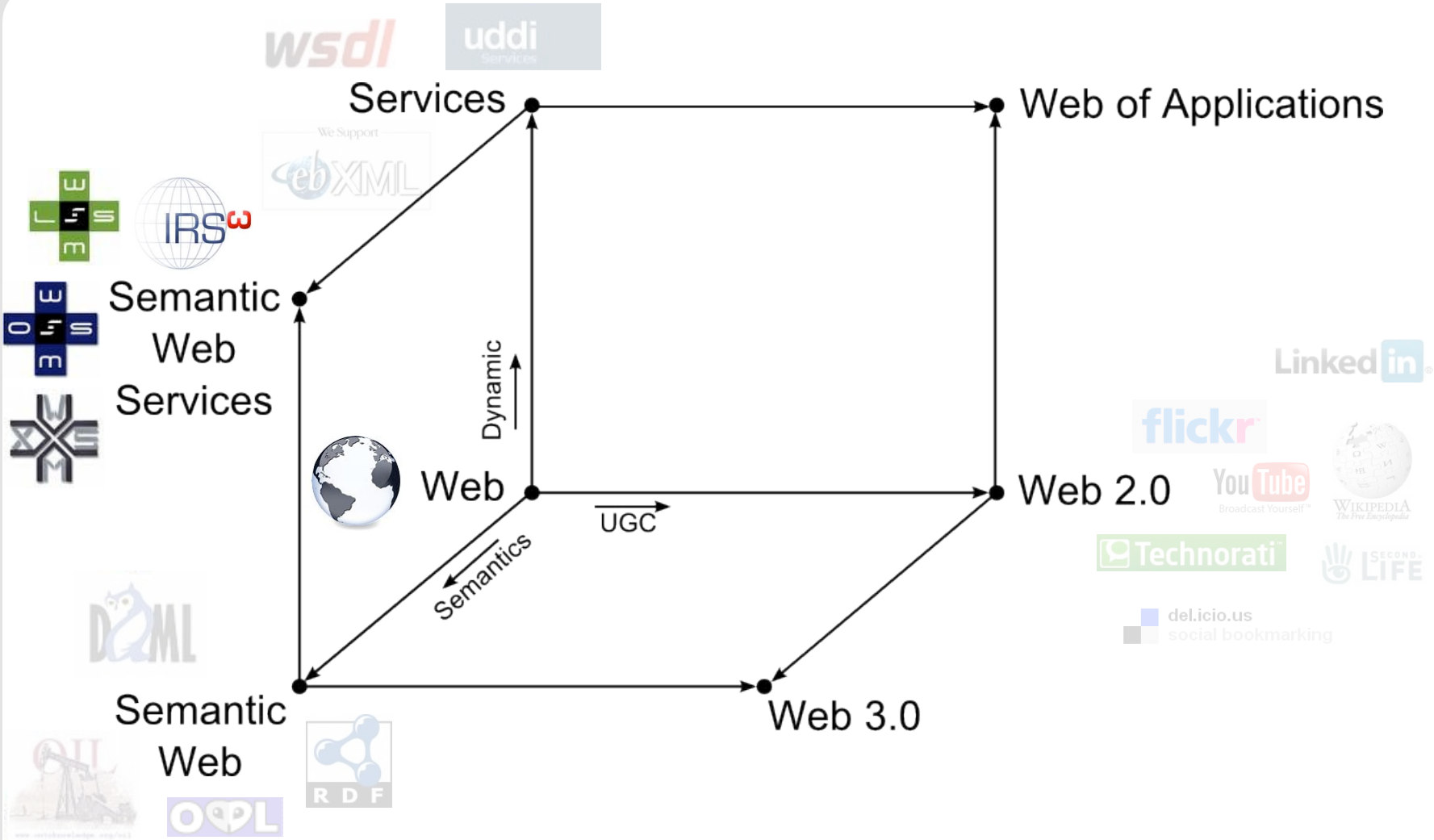
The Semantic Web



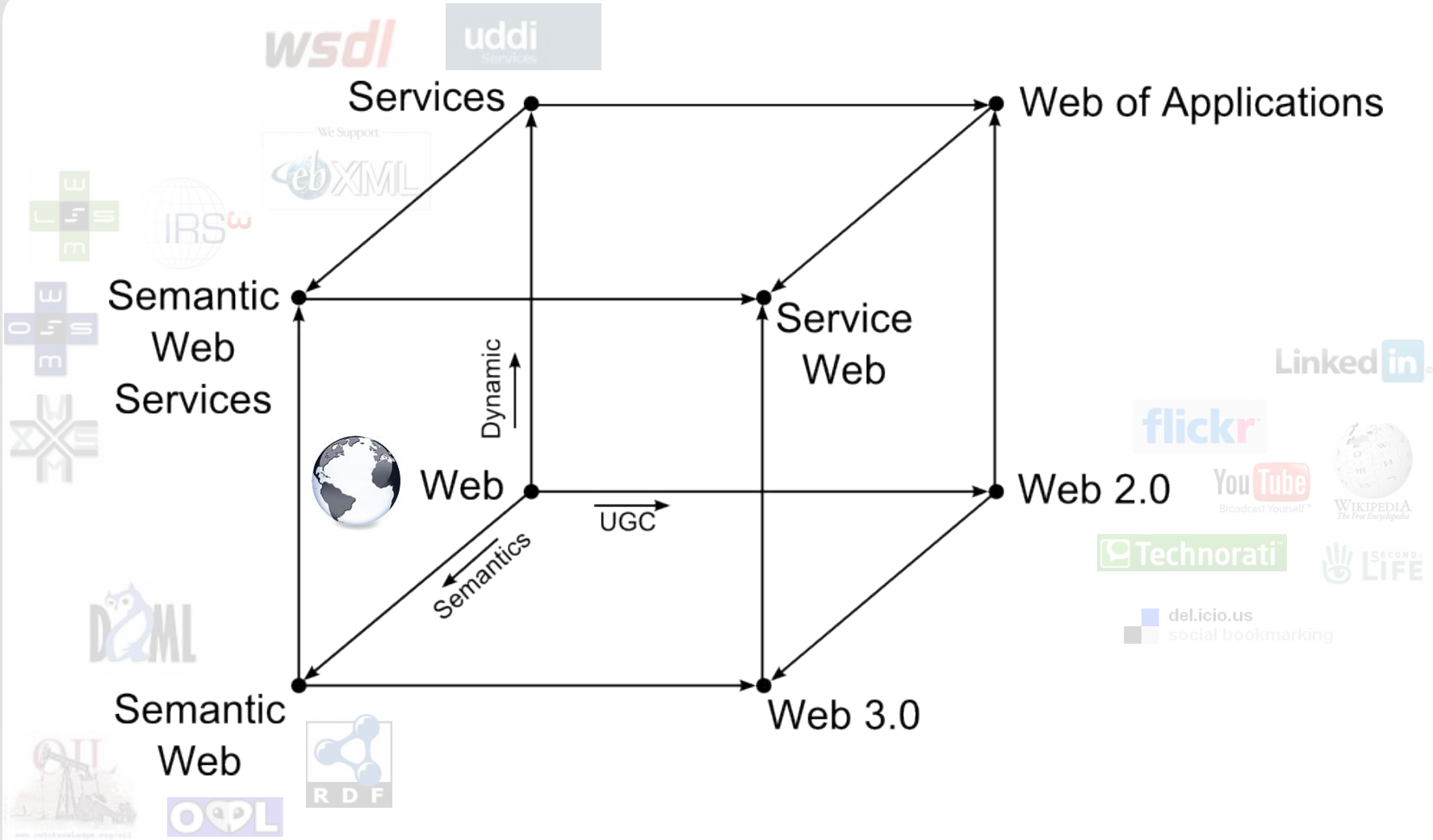
Web 3.0


wsdl
uddi
Services
Web of Applications
We Support
ebXML
Dynamic
Web
Web 2.0
UGC
Semantics
Semantic Web
Web 3.0
RDF
OIL
OWL
DAML
Linked in
flickr
YouTube
Broadcast Yourself
WIKIPEDIA
Technorati
SECOND LIFE
del.icio.us
social bookmarking

Semantic Web Services



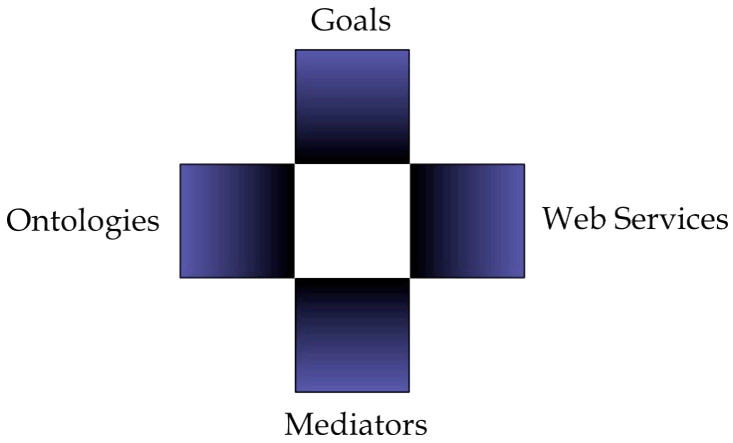
Service Web


Services
Web of Applications
Semantic Web Services
Service Web
Dynamic
Web
Web 2.0
UGC
Semantics
Semantic Web
Web 3.0

WSMO Recap

Objectives that a client wants to achieve by using Web Services

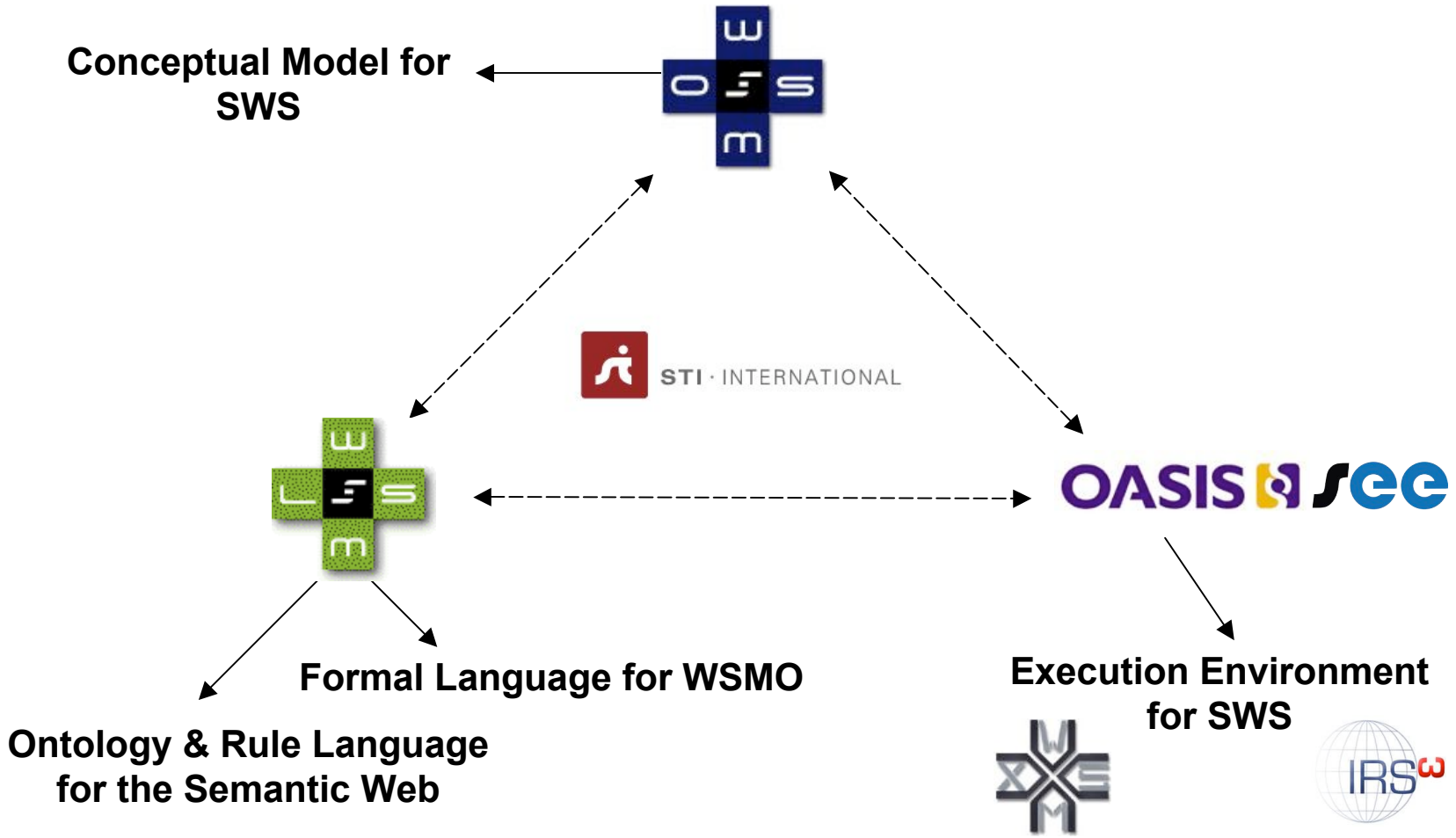
Formally specified terminology used by all other components



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

Connectors between components with mediation facilities for handling heterogeneities

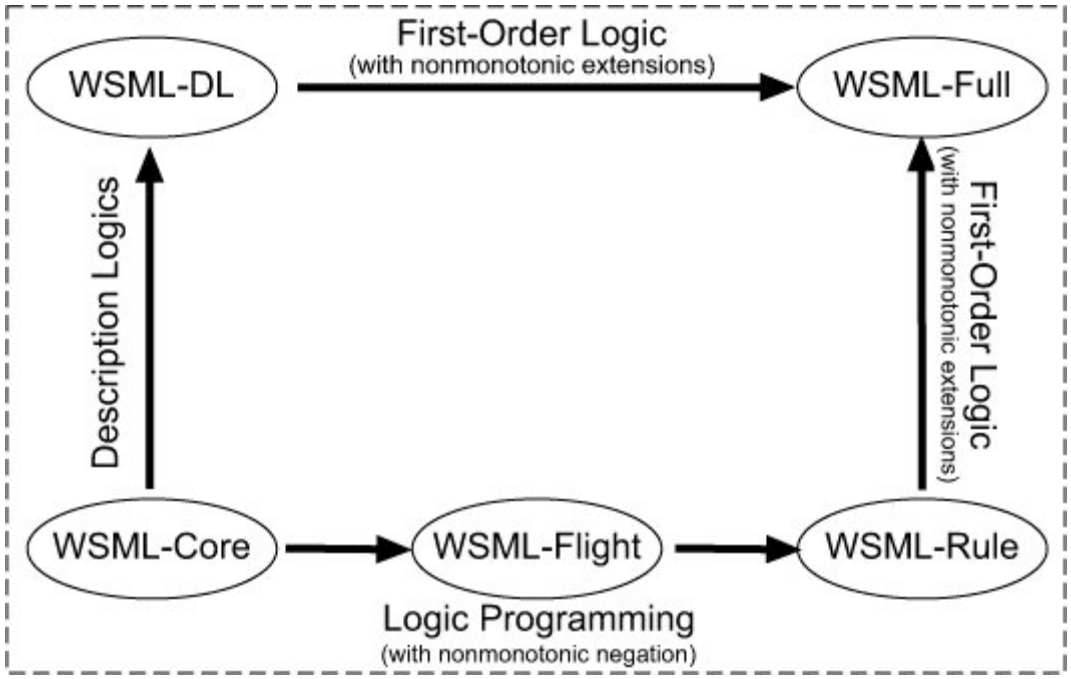
WSMO, WSML and SEE

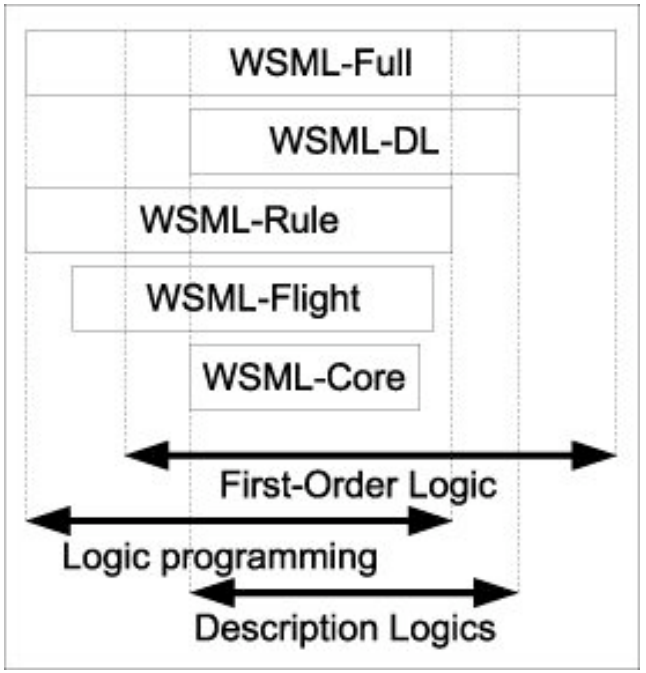


WSML

- 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.wsmo.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
    hasActor ofType actor

instance dan_castellanata memberOf actor
    hasName hasValue "Dan Castellaneta"

instance homer_simpson memberOf character
    hasName hasValue "Homer Simpson"
    hasActor hasValue dan_castellanata
```

Human readable Syntax

```
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 amazonChorography
        …
    orchestration amazonOrchestration
        …
```

Reasoning with WSML

- 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

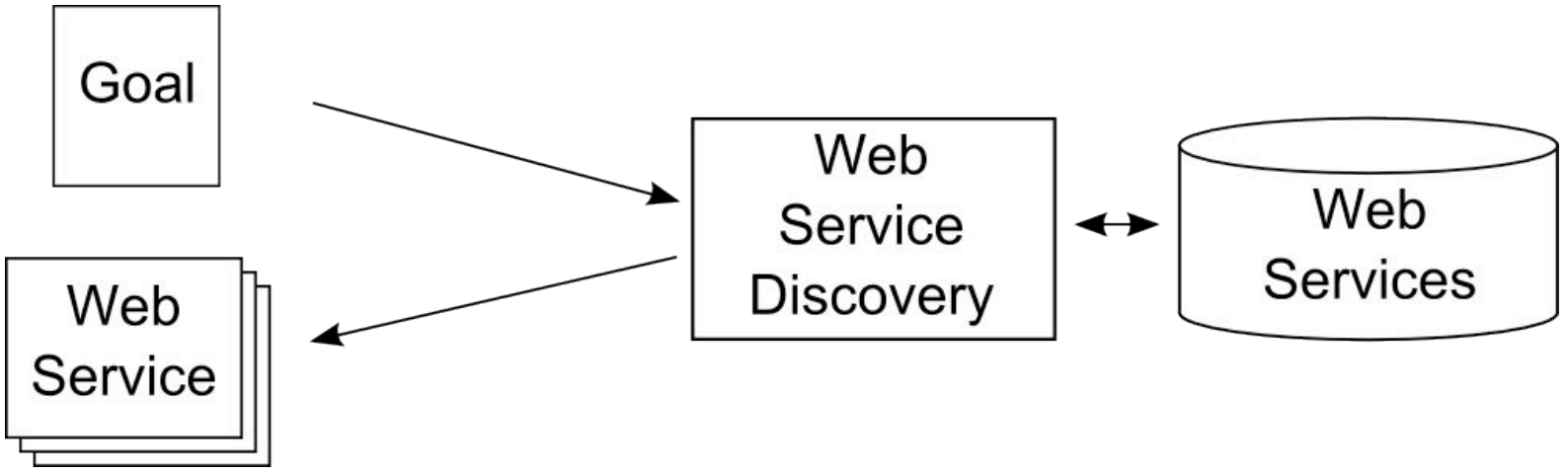
Semantic Execution Environments

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

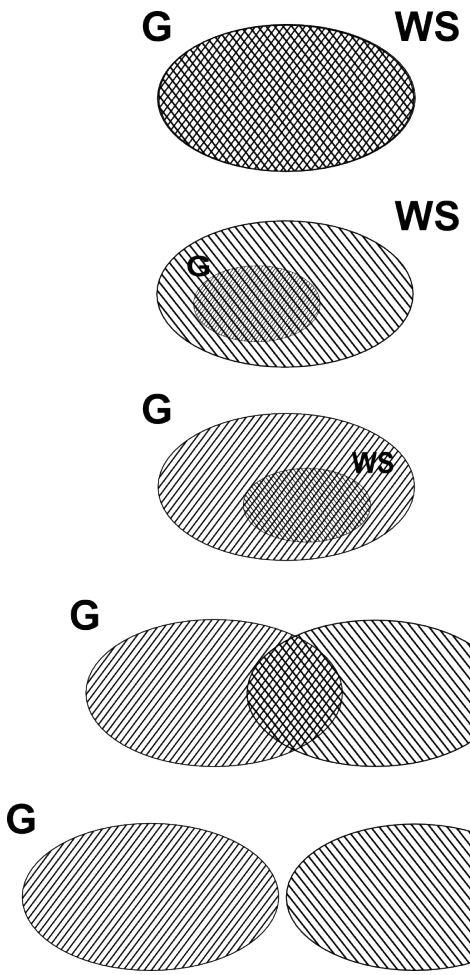
- Enable the automation of previously human intensive tasks when building applications with a Service Oriented Architecture
  - Discovery: Determine usable services for a request
  - Composition: Combine services to achieve a goal
  - Ranking and Selection: Choose appropriate service for the job
  - Mediation: Solve mismatches to enable interoperability
  - Invocation: Execute entry points on the service

Discovery

- Find Semantic Web Services that can totally or partial fulfil the end users Goal

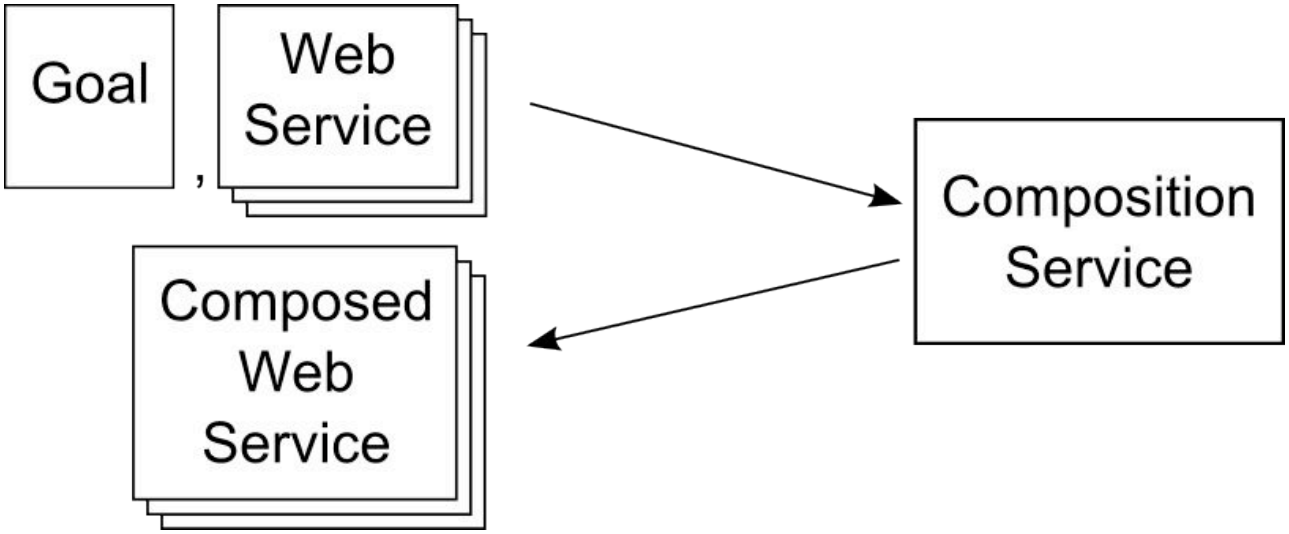


Discovery


G
WS
Exact Match
WS
G
Plugin Match
G
WS
Subsumption Match
G
WS
Intersection Match
G
WS
No Match

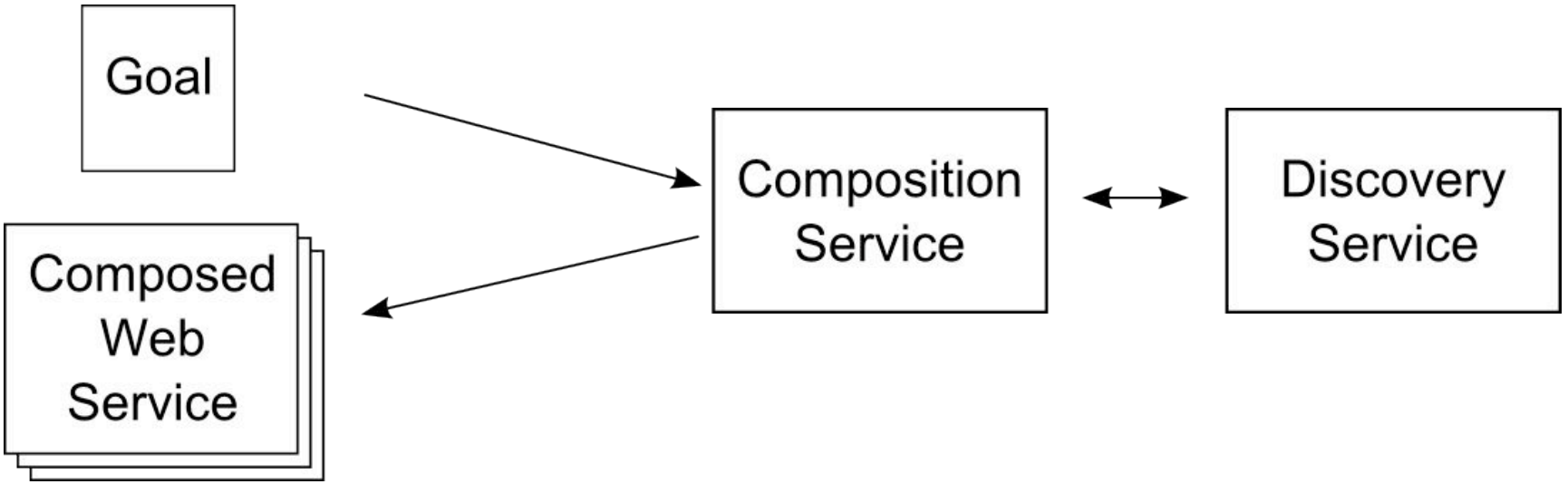
Composition

- Combine a number of Semantic Web Services together to fulfil the end users Goal



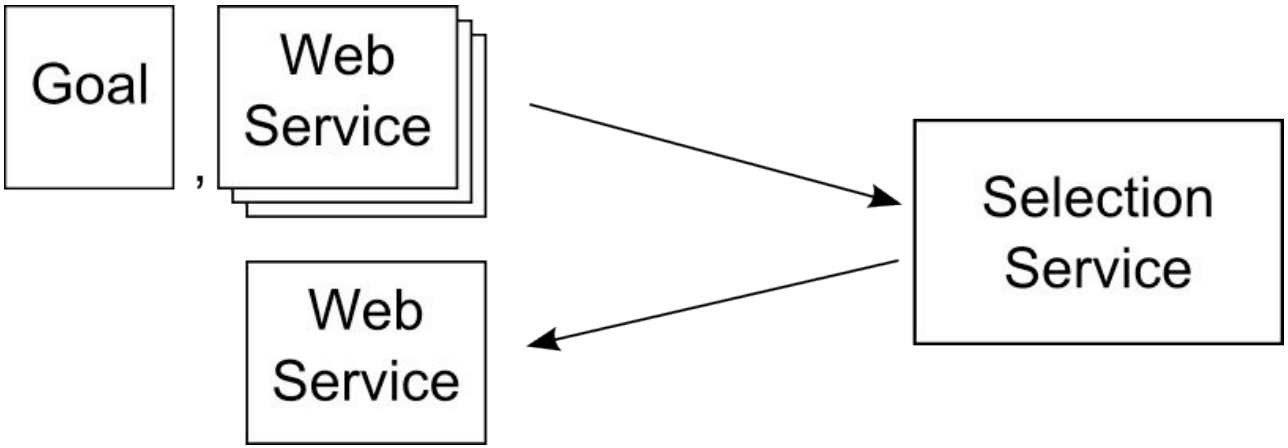
Composition

- Combine a number of Semantic Web Services together to fulfil the end users Goal



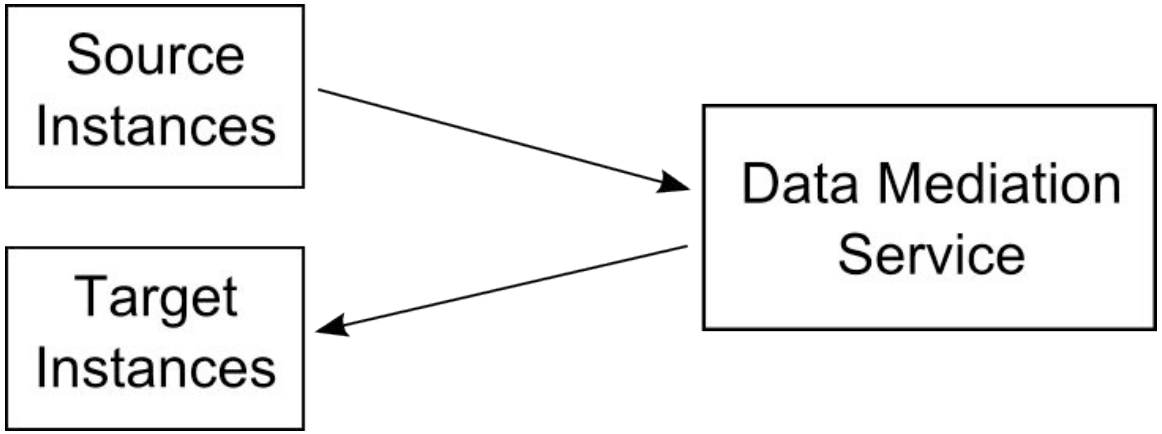
Ranking and Selection

- Choosing the most appropriate Web Service that meets the end users non-functional requirements



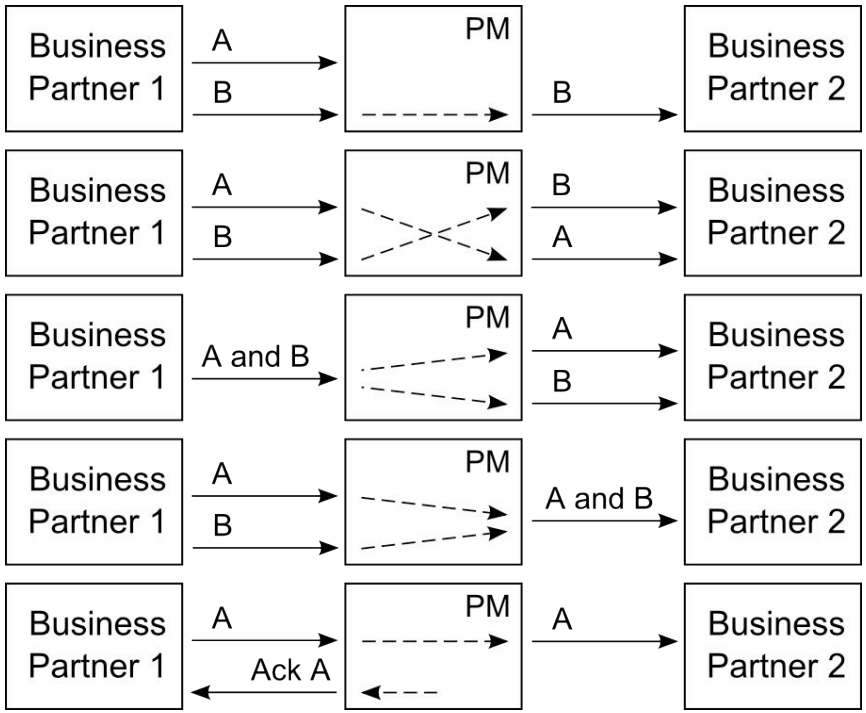
Mediation

- Data Mediation for resolving terminological mismatches and enabling interoperability at the data level
  - Ontology Merging
  - Ontology Alignment
  - Instance Transformation



Mediation

- 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

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)

WSML

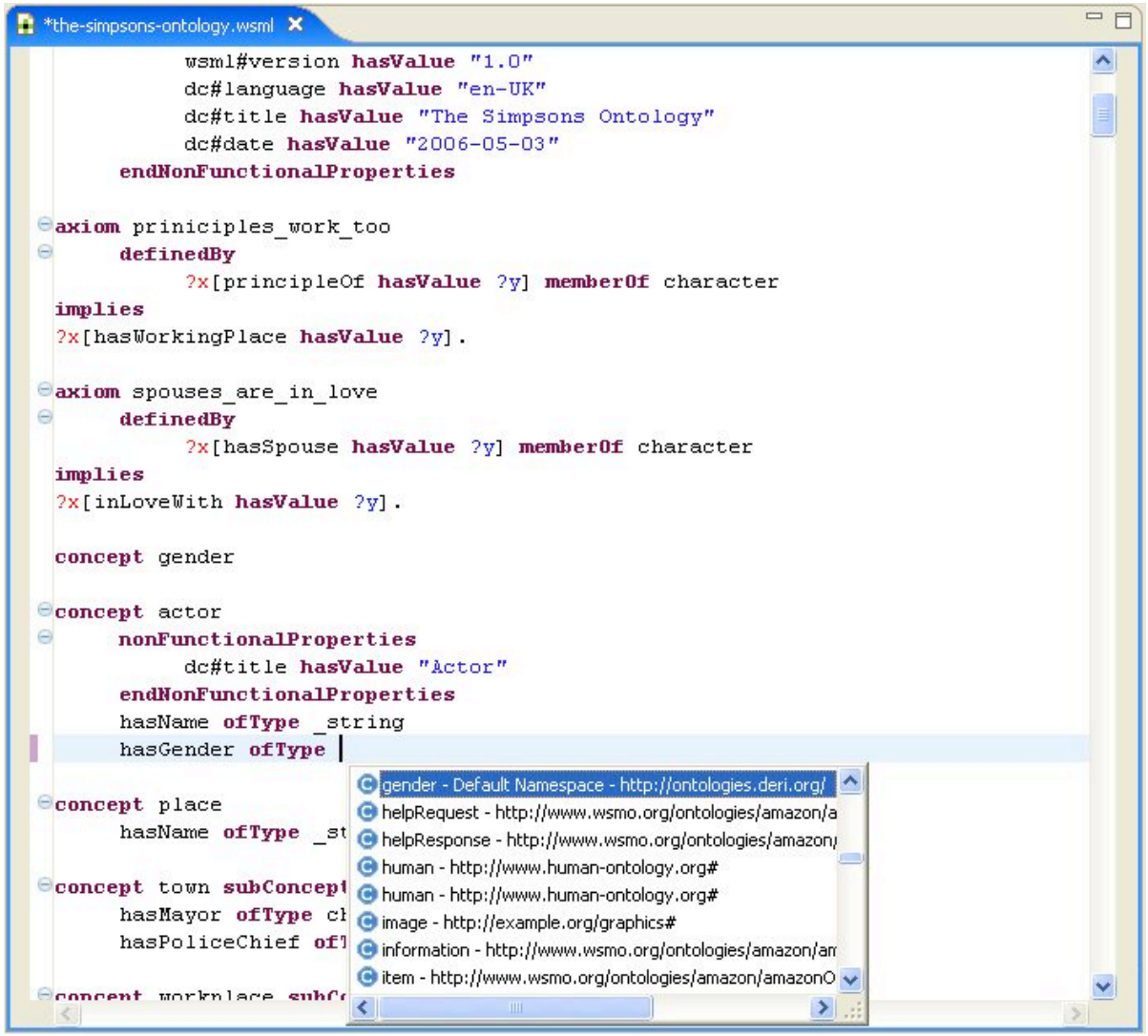
- Semantic Execution Environments need Ontologies, Goals, Web Services, and Mediators in order to function
- Provide support to the developer in creating these descriptions
- Provide mechanisms for browsing semantic descriptions to aid in developer understanding
- Abstract the developer from the underlying syntax
- Assist in the validation and testing of semantic descriptions

WSML Text Editor

- 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

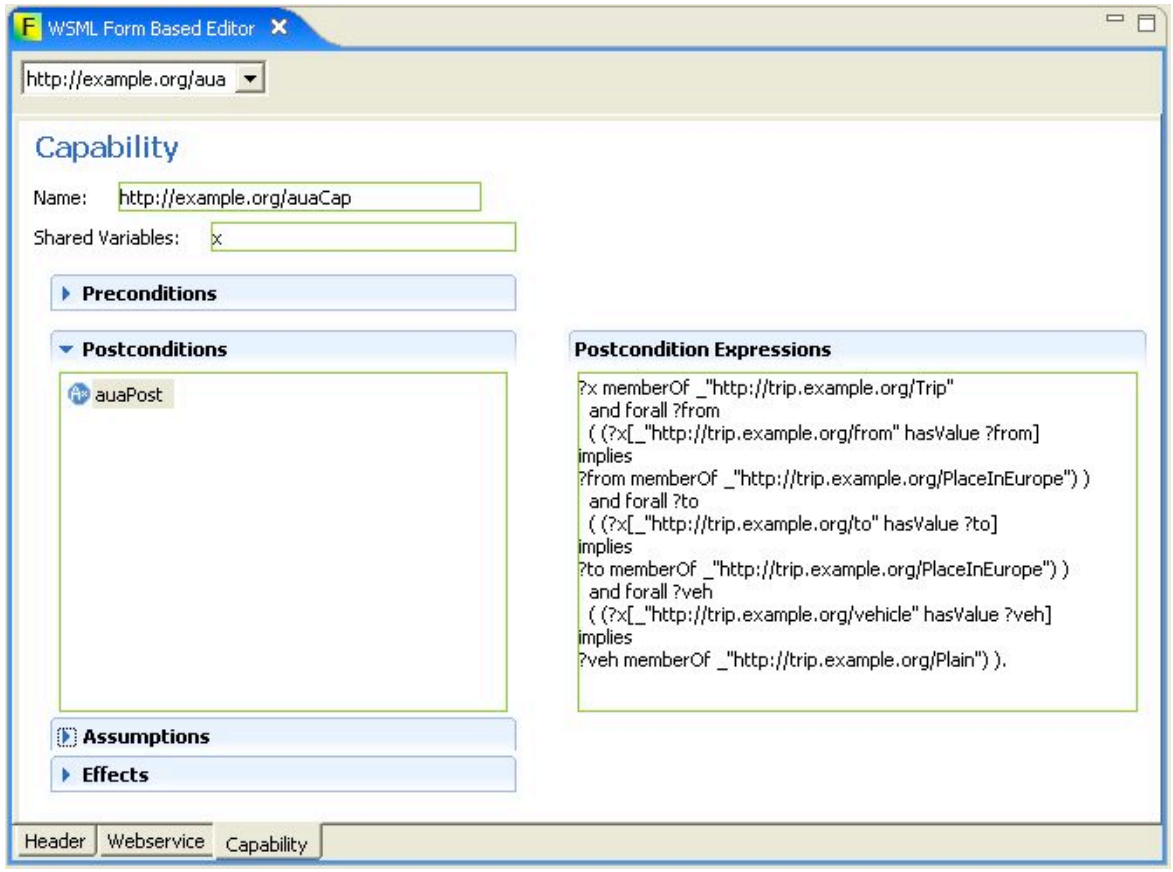
Content Folding

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

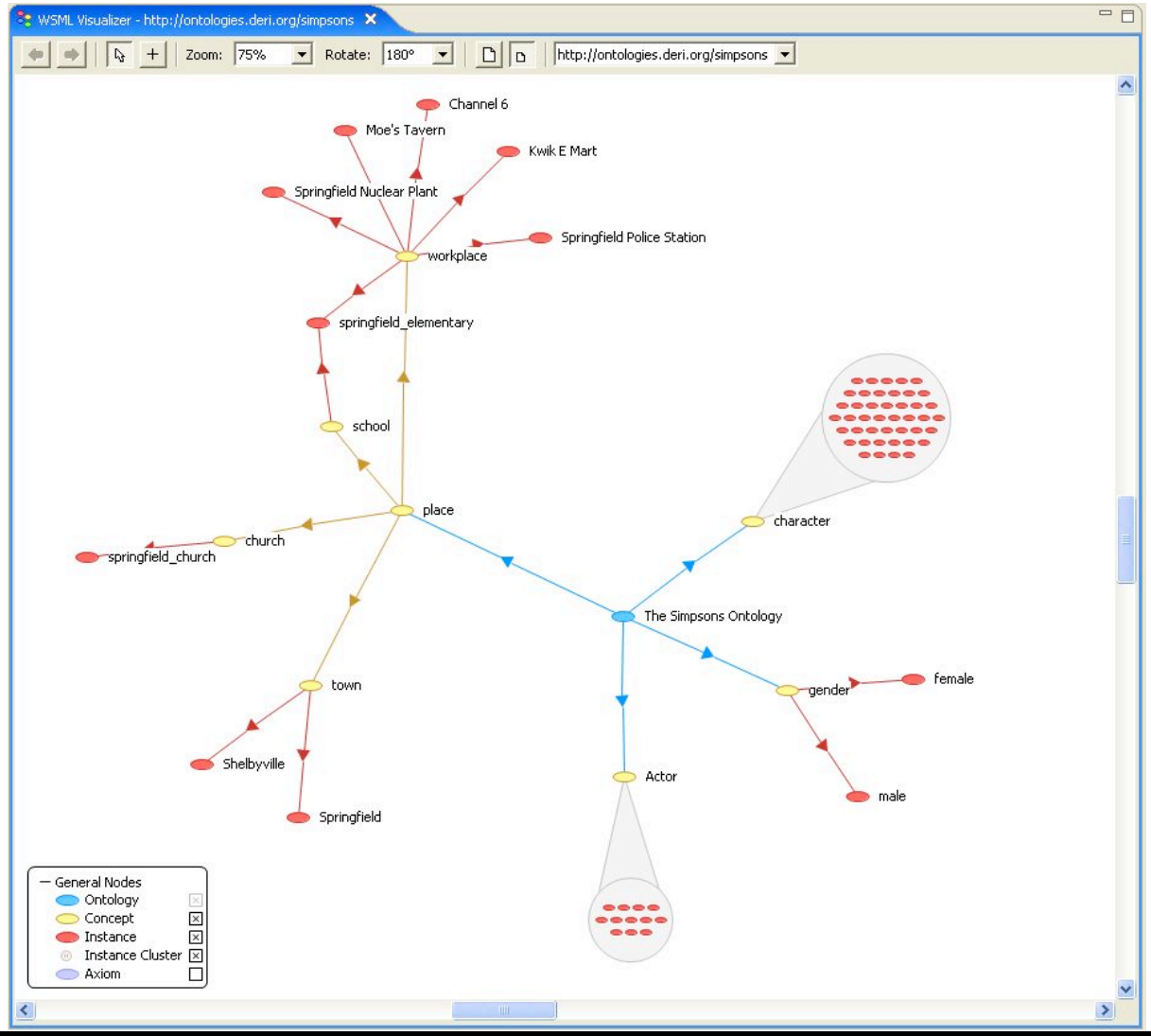
WSML Form based Editor



WSML Visualizer

- 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

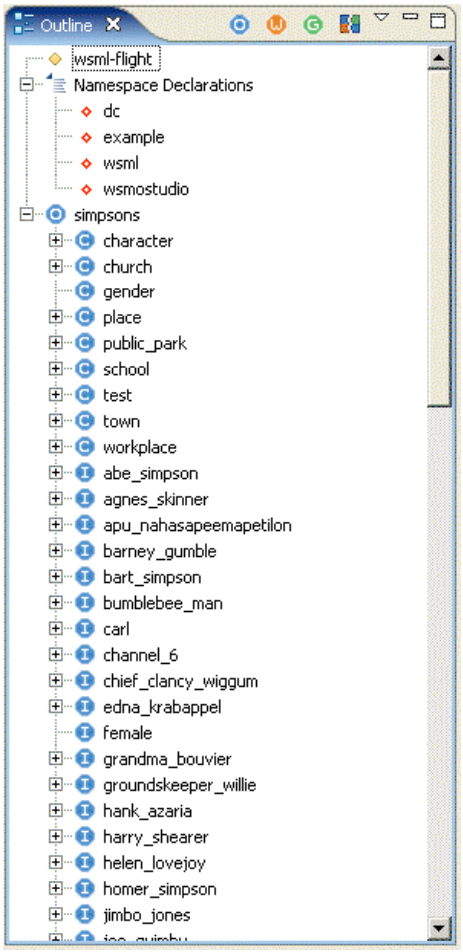
Semantic Levels

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

Outline View

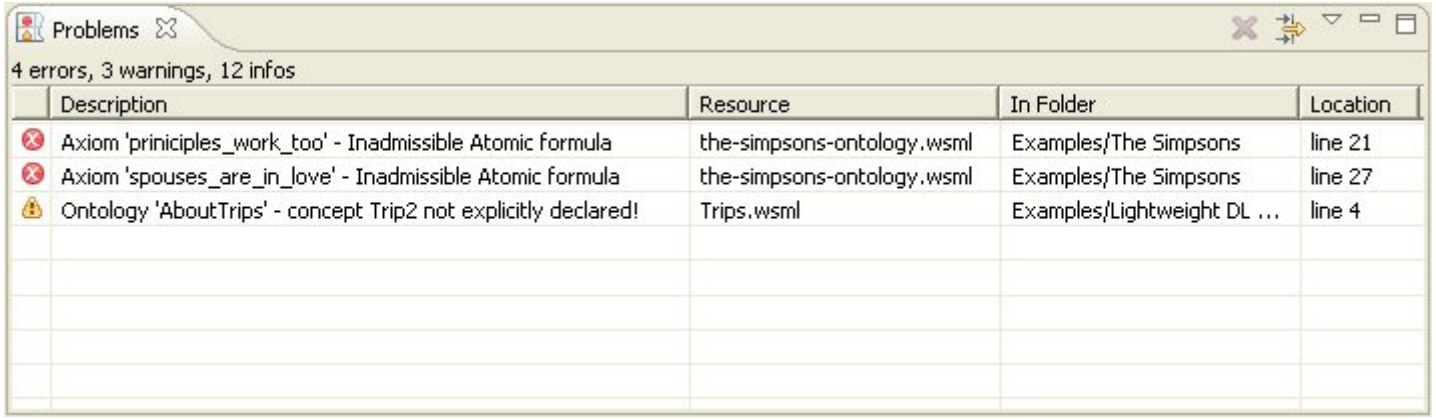


Validation

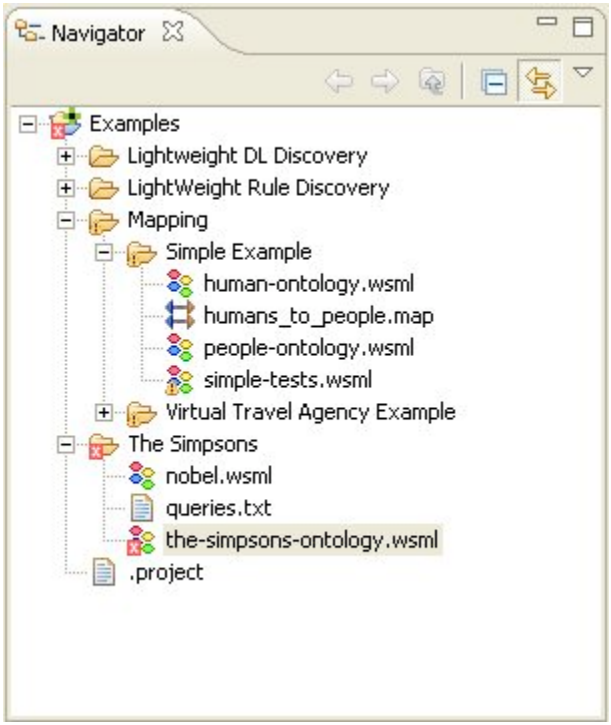
- WSMO4J parser used to validate syntax
- WSMO4J validator used to validate semantics
  - Ensures features within the semantic description match that of the specified WSML Variant (Errors)
  - Checks for unrecommended usage of WSML Features (Warnings)

- 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



WSML Navigator



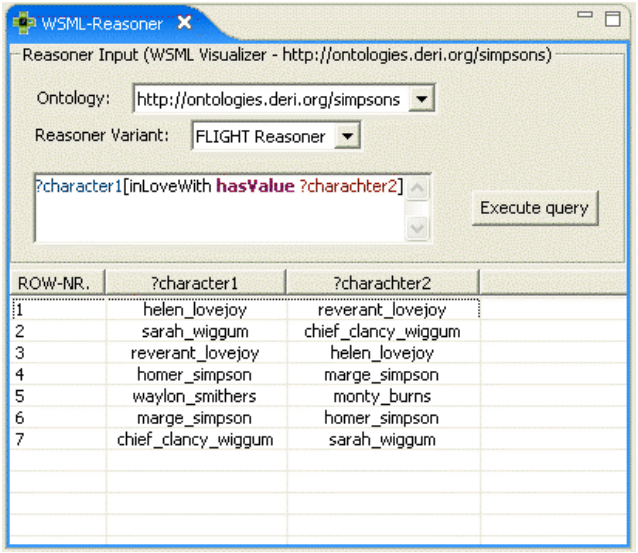
Testing

- 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

WSML Reasoning View

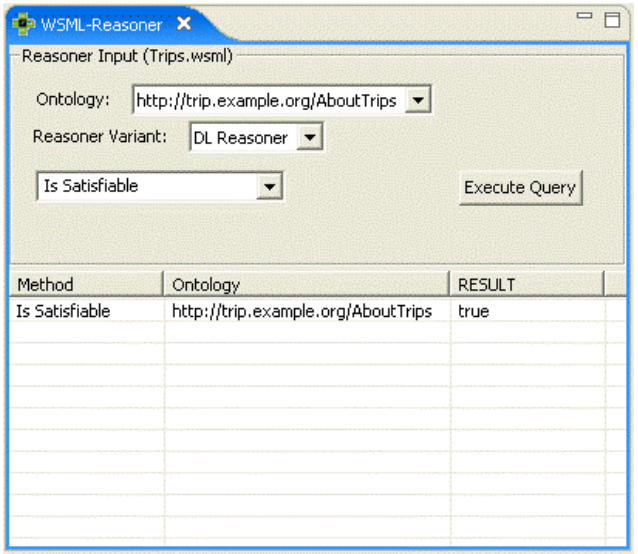


Ontology selection

Reasoner selection

Syntax Highlighting

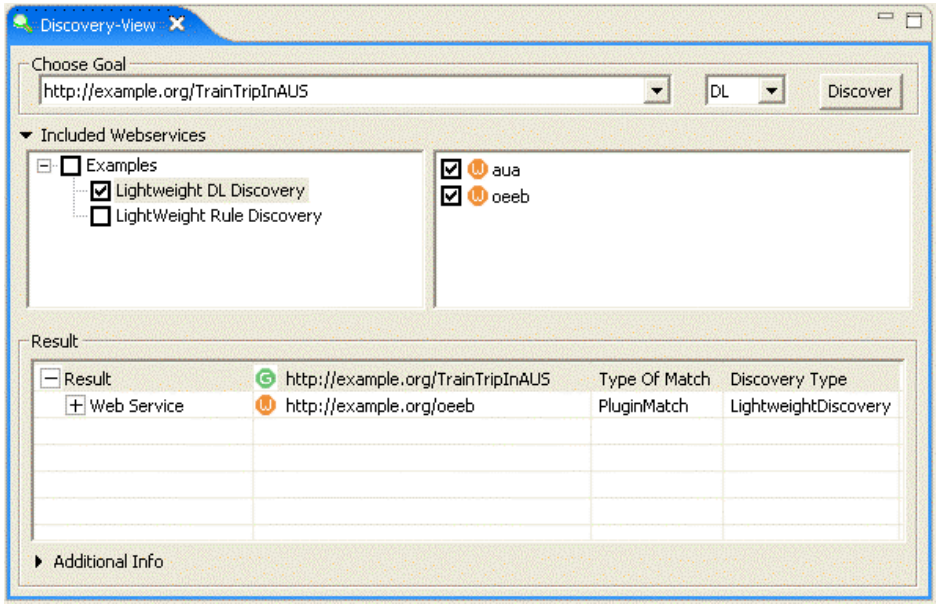
Interfacing with editors



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

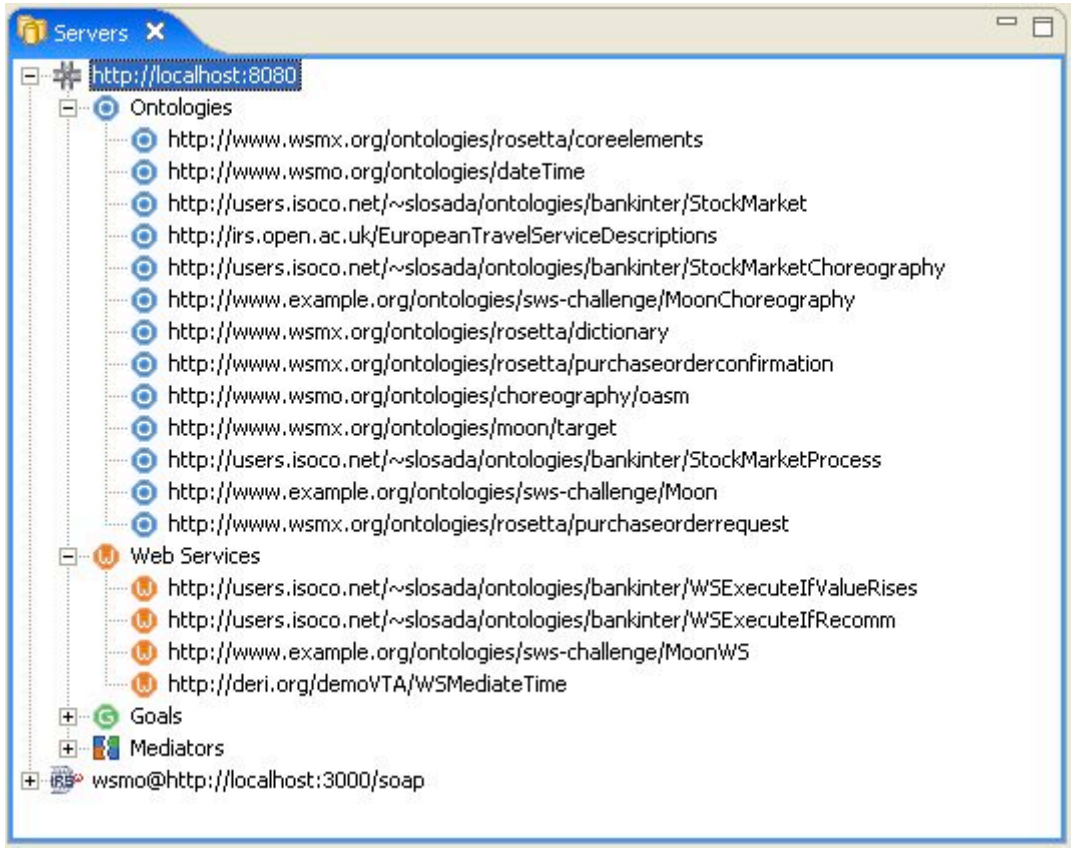
Type of Match

Interfacing with editors

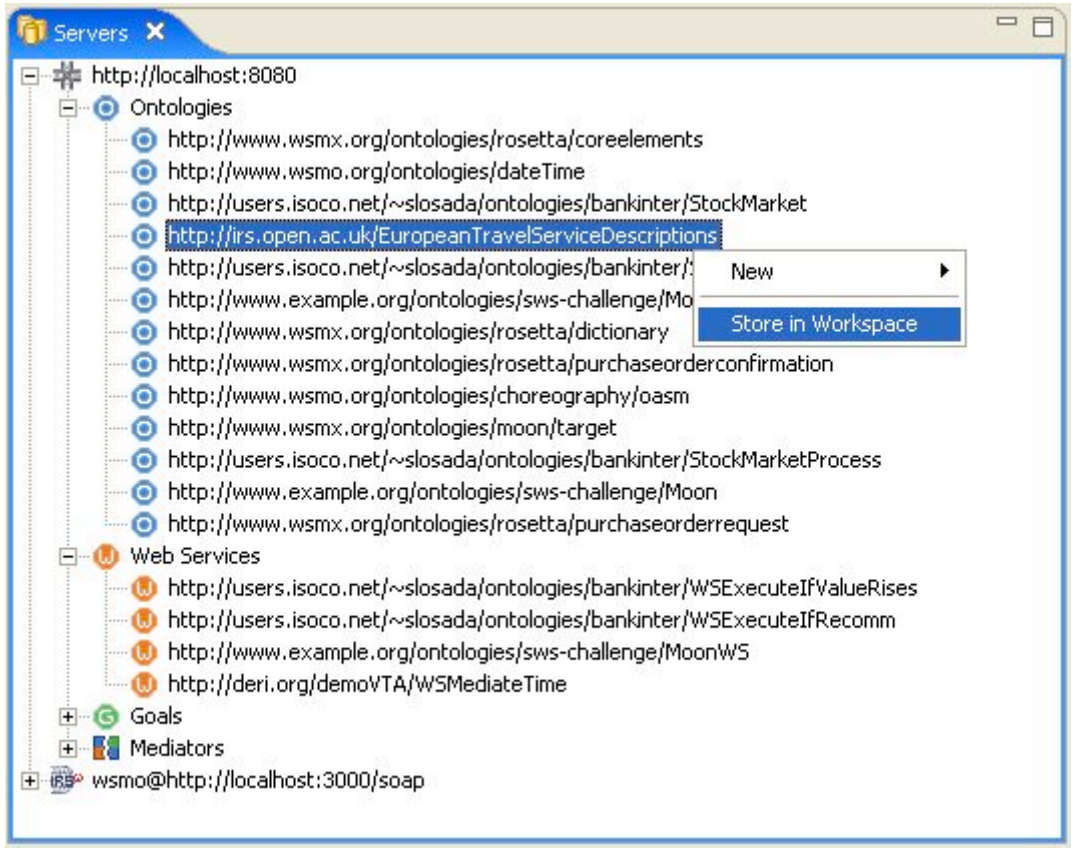
Interfacing with a SEE

- 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

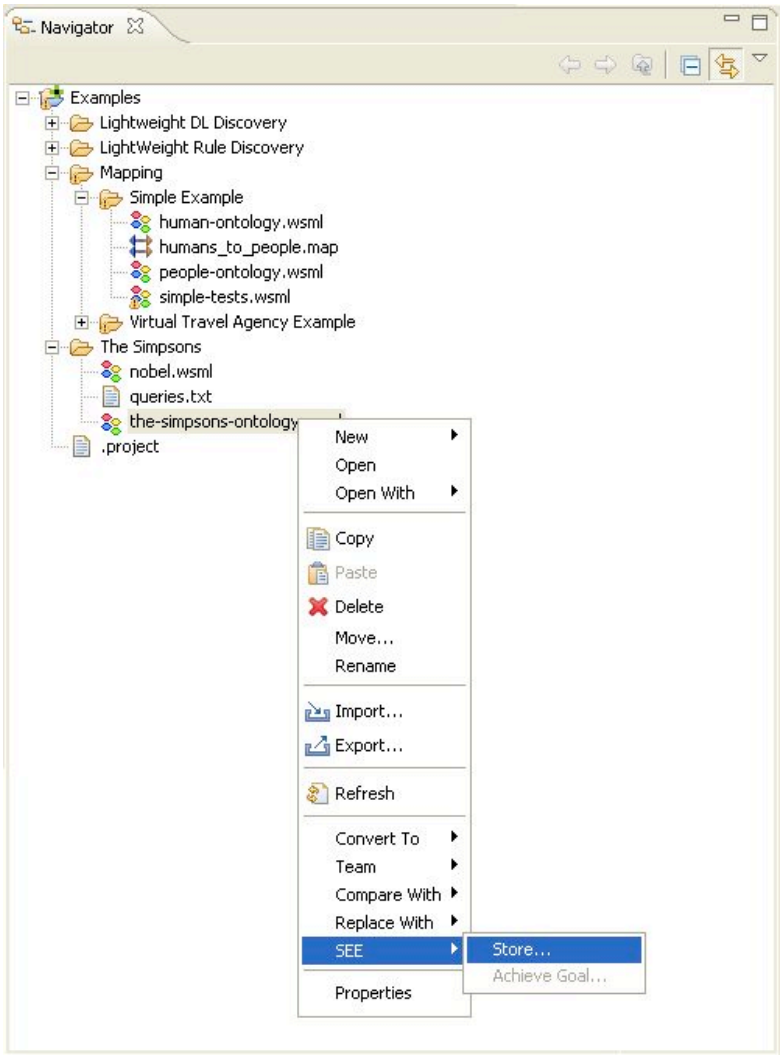
Browsing WSML in a SEE



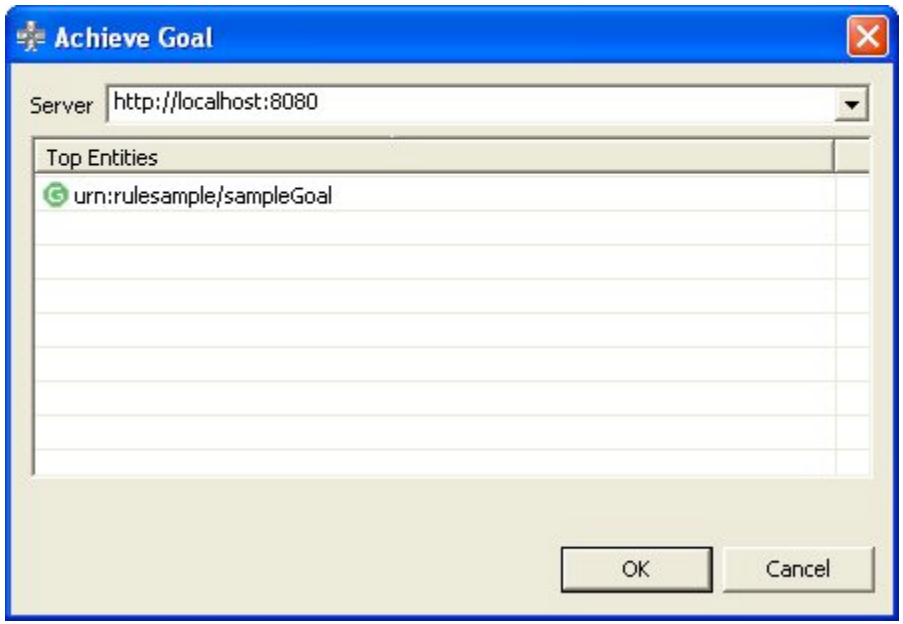
Retrieving WSML from a SEE



Storing WSML to a SEE



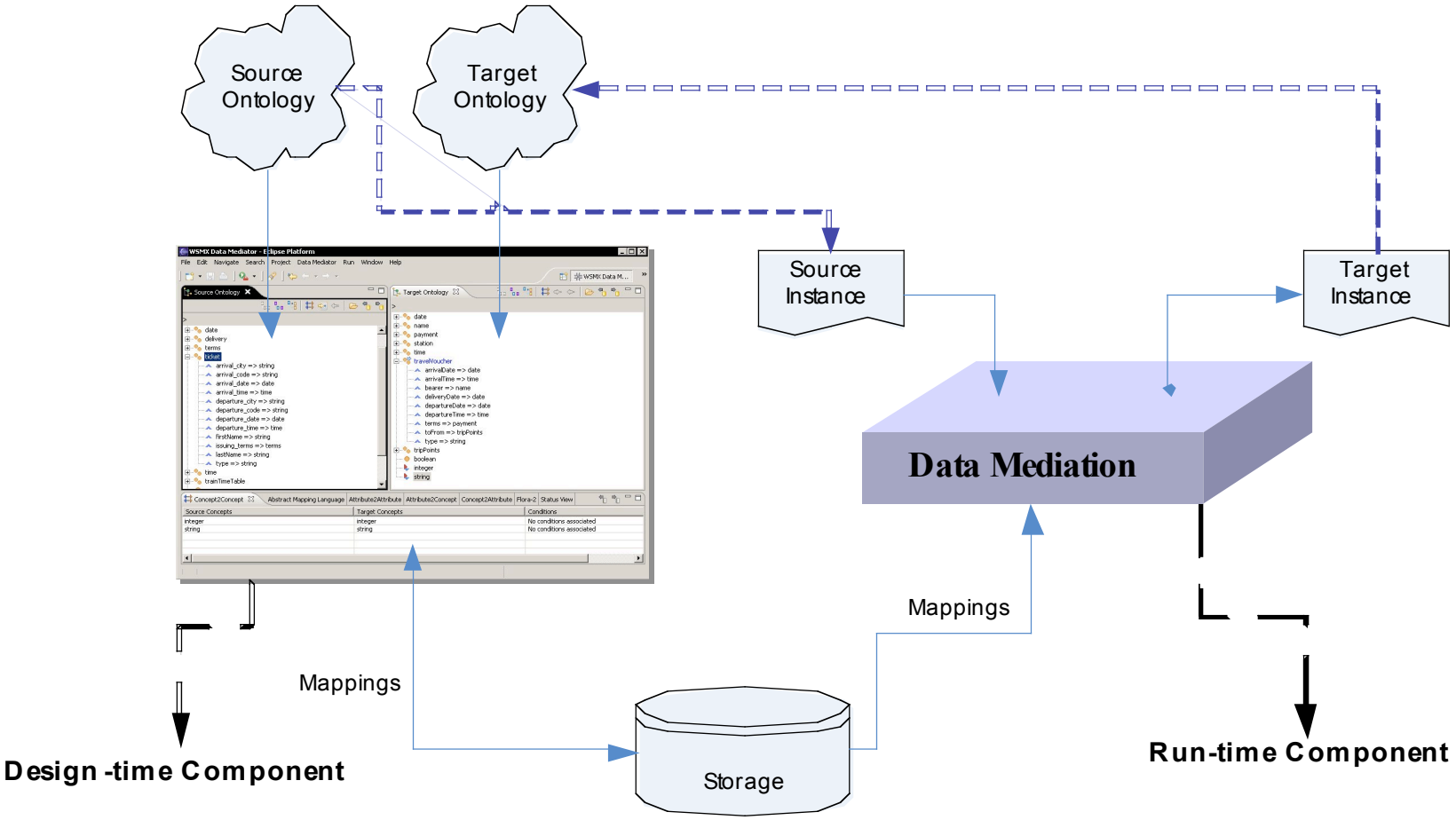
Invoking a SEE



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

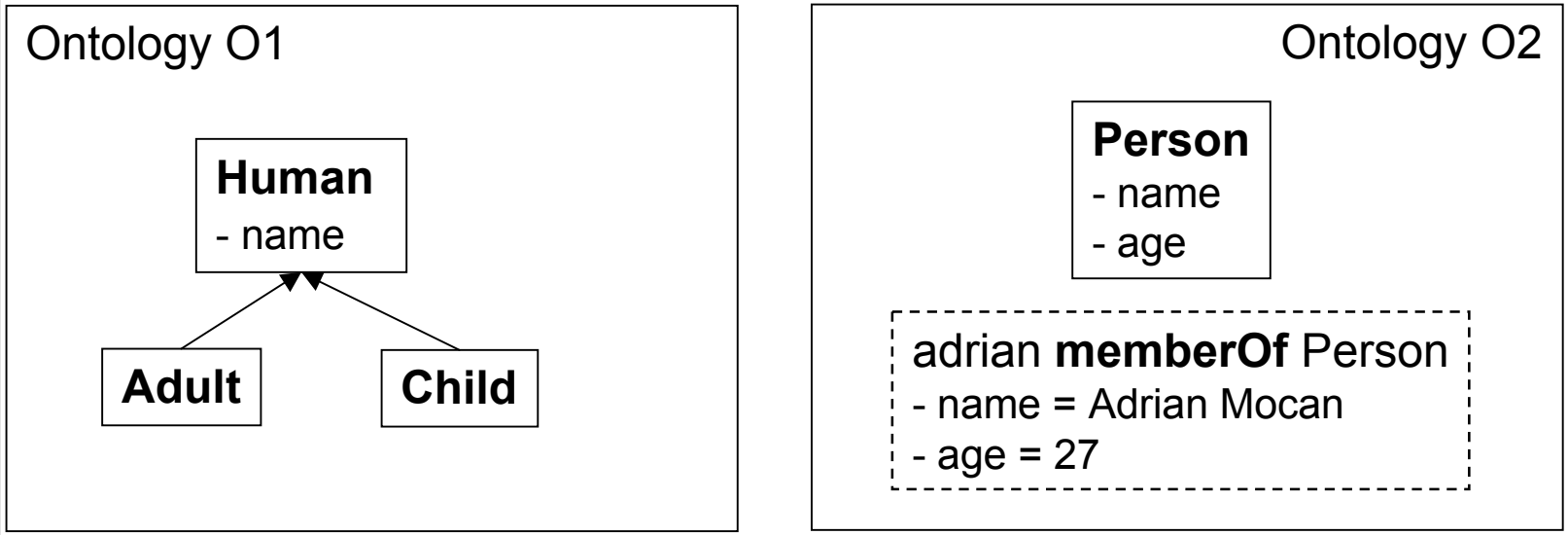
Ontology Mediation (WSMX/WSMT)



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

Mapping Language Example



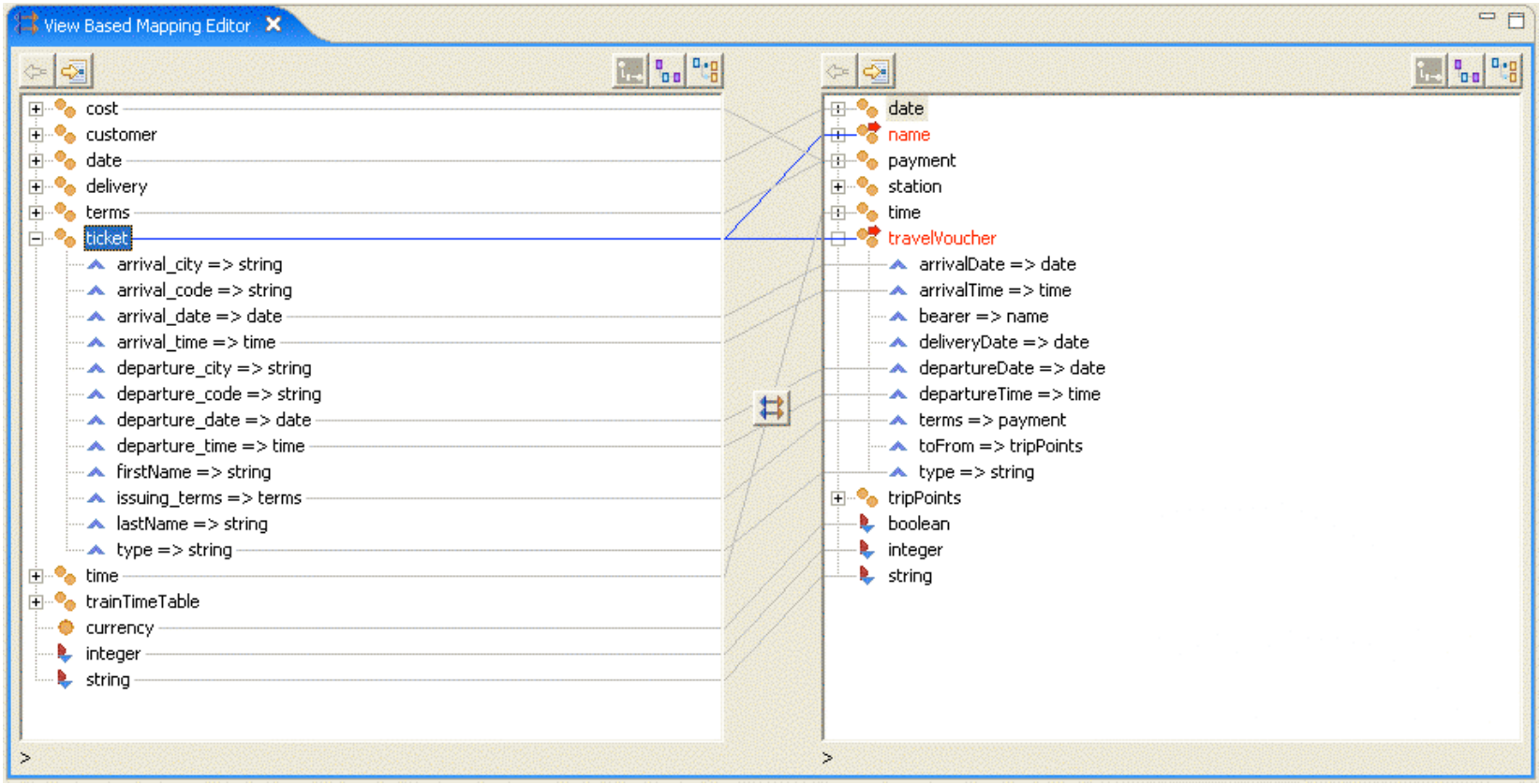
classMapping(**unidirectional** o2:Person o1.Adult
attributeValueCondition(o2.Person.age **>=** 18))

This allows to transform the instance ‘adrian’ of concept person in ontology O2 into a valid instance of concept ‘adult’ in ontology O1

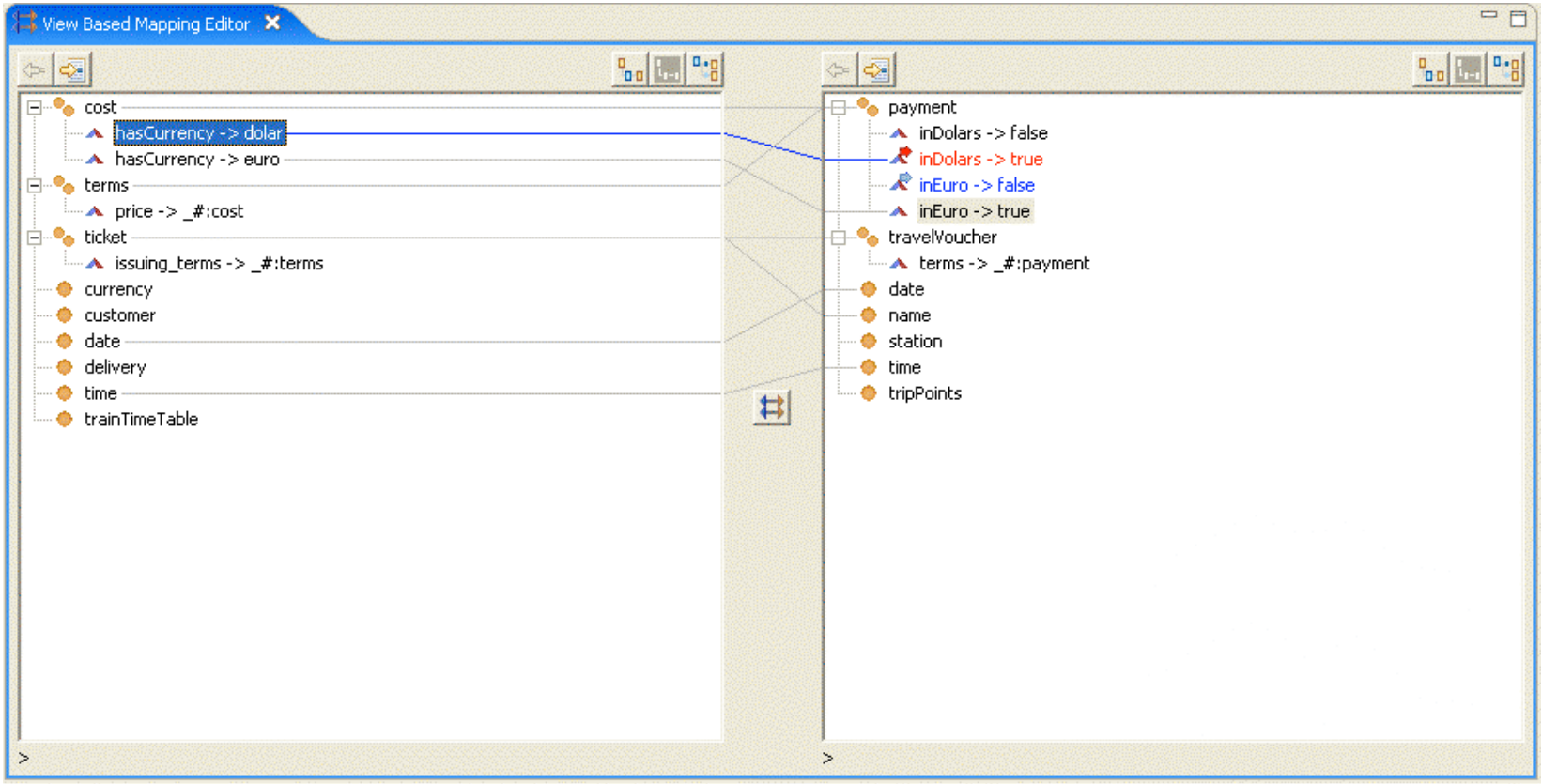
View based Editor

- 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

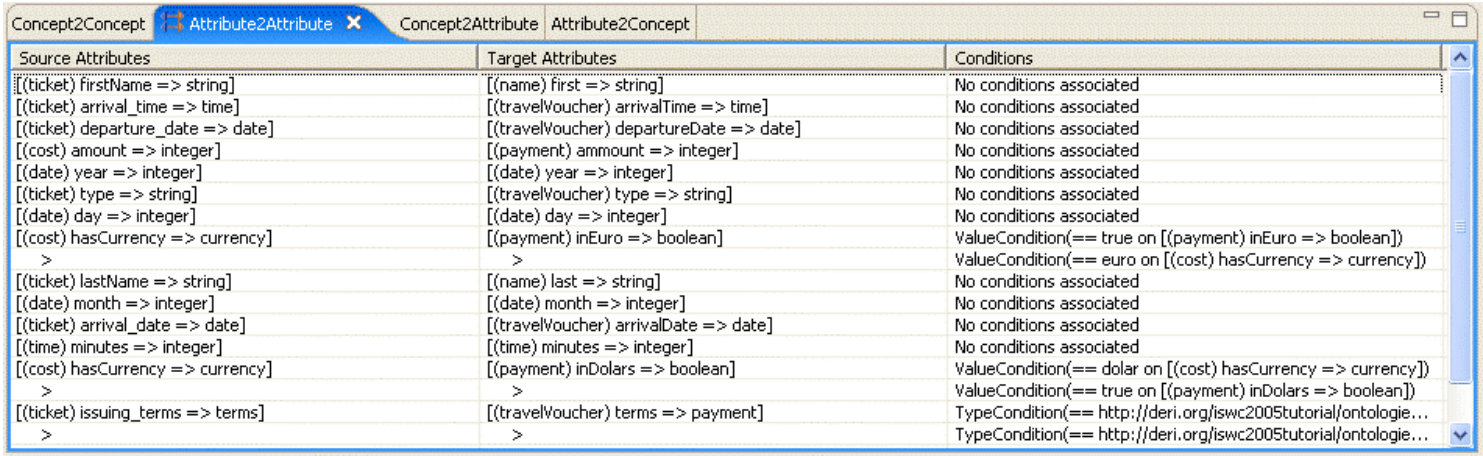
- As ontologies become bigger mappings can be harder to see
- View based editor also obscures the type of the mapping
- Provide the developer with a mechanism for quickly seeing mappings by type
- Provide a mechanism for deleting one or more mappings

Mapping Views (Concept2Concept)

Concept2Concept | Attribute2Attribute | Concept2Attribute | Attribute2Concept

Source Concepts	Target Concepts	Conditions
time	time	No conditions associated
ticket	travelVoucher	No conditions associated
currency	boolean	No conditions associated
string	string	No conditions associated
integer	integer	No conditions associated
terms	payment	No conditions associated
date	date	No conditions associated
ticket	name	No conditions associated
cost	payment	No conditions associated

Mapping Views (Attribute2Attribute)



Concept2Concept | Attribute2Attribute | Concept2Attribute | Attribute2Concept

Source Attributes	Target Attributes	Conditions
[(ticket) firstName => string]	[(name) first => string]	No conditions associated
[(ticket) arrival_time => time]	[(travelVoucher) arrivalTime => time]	No conditions associated
[(ticket) departure_date => date]	[(travelVoucher) departureDate => date]	No conditions associated
[(cost) amount => integer]	[(payment) ammount => integer]	No conditions associated
[(date) year => integer]	[(date) year => integer]	No conditions associated
[(ticket) type => string]	[(travelVoucher) type => string]	No conditions associated
[(date) day => integer]	[(date) day => integer]	No conditions associated
[(cost) hasCurrency => currency]	[(payment) inEuro => boolean]	ValueCondition(== true on [(payment) inEuro => boolean])
>	>	ValueCondition(== euro on [(cost) hasCurrency => currency])
[(ticket) lastName => string]	[(name) last => string]	No conditions associated
[(date) month => integer]	[(date) month => integer]	No conditions associated
[(ticket) arrival_date => date]	[(travelVoucher) arrivalDate => date]	No conditions associated
[(time) minutes => integer]	[(time) minutes => integer]	No conditions associated
[(cost) hasCurrency => currency]	[(payment) inDolars => boolean]	ValueCondition(== dolar on [(cost) hasCurrency => currency])
>	>	ValueCondition(== true on [(payment) inDolars => boolean])
[(ticket) issuing_terms => terms]	[(travelVoucher) terms => payment]	TypeCondition(== http://deri.org/iswc2005tutorial/ontologie…
>	>	TypeCondition(== http://deri.org/iswc2005tutorial/ontologie…

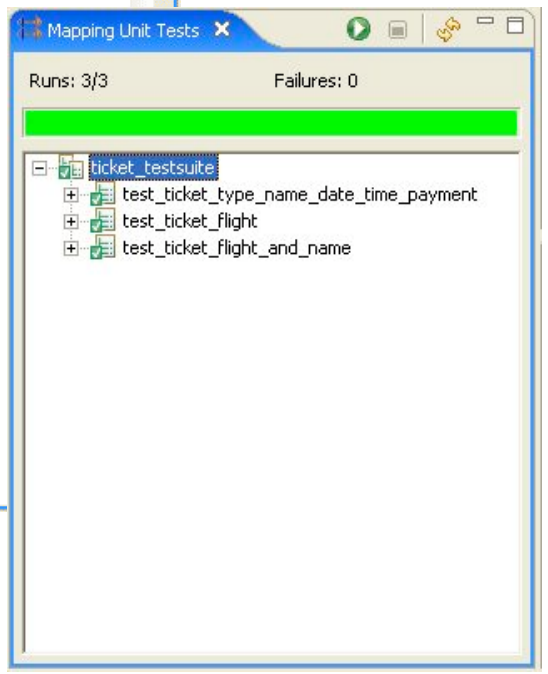
Testing

- 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

Mapping Unit Test View

tests.wsml

```
//INPUT

instance my_ticket_1 memberOf travel1#ticket
     travel1#type hasValue "flight"

//EXPECTED OUTPUT
instance expected_travelVoucher_1 memberOf travel2#travelVoucher
    travel2#type hasValue "flight"

/*-------------------------------------------------------------------------------*/

instance test_ticket_flight_and_name memberOf munit#test
     munit#hasSourceInstance hasValue my_ticket_2
     munit#hasTargetInstance hasValue expected_travelVoucher_2

//INPUT

instance my_ticket_2 memberOf travel1#ticket
     travel1#type hasValue "flight"
     travel1#firstName hasValue "Adrian"
     travel1#lastName hasValue "Mocan"

//OUTPUT

instance expected_travelVoucher_2 memberOf travel2#travelVoucher
    travel2#bearer hasValue expected_name_2
    travel2#type hasValue "flight"

instance expected_name_2 memberOf travel2#name
    travel2#last hasValue "Mocan"
    travel2#first hasValue "Adrian"
```



Questions?

Questions?

… and then lets use the toolkit!