

WSMO Studio Users Guide
v. 1.8

Marin Dimitrov
marin.dimitrov@ontotext.com

Alex Simov
alex.simov@ontotext.com

Vassil Momtchev
vassil.momtchev@ontotext.com

Mihail Konstantinov
mihail.konstantinov@ontotext.com

May 11, 2006

Contents

1	Introduction	9
2	Installing <i>WSMO Studio</i>	11
2.1	Requirements	11
2.2	Licence	11
2.3	Installation Types	12
2.3.1	Standalone Mode	12
2.3.2	Embedded Mode	12
3	Introduction to Eclipse	13
3.1	Plug-in Based Architecture	13
3.1.1	Runtime Kernel	13
3.1.2	Plug-ins	14
3.2	User Interface	15
4	<i>WSMO Studio</i> UI	17
4.1	Menus	17
4.2	Wizards	18
4.3	Preferences	18
4.4	Perspectives	19
4.5	Project Navigator	21

5	The WSMO Editor	23
5.1	WSMO Navigator	23
5.2	Editors	24
5.2.1	Common Editors	25
5.2.2	Ontology Editors	25
5.2.3	Concept / Relation Editor	26
5.2.4	Instance Editor	28
5.2.5	Axiom Editor	28
5.2.6	Goal / Web Service Editors	28
5.2.7	Mediator Editors	29
5.3	WSML Text Editor	30
6	Choreography Editor	31
6.1	WSMO Based Choreography	31
6.2	User Interface	32
6.2.1	State Signature Editor	33
6.2.2	Transition Rules Editor	34
7	Working with Repositories	37
7.1	Repository Perspective	37
7.1.1	Repository Explorer	37
7.1.2	Repository Views	39
7.1.3	Import and Export	39
7.2	Integrated ORDI Repository	40
8	WSML Validator	41
9	WSML Reasoner	43

9.1	Installation	43
9.2	Usage	44
10	3rd Party Plug-ins	46
10.1	IRS-III Adapter	46
10.2	Infrawebs Axiom Editor	46
10.3	WSMO Visualiser	48
10.3.1	Installation	48
10.3.2	Usage	49
11	Using the Update Site	50
12	User Support	53
	References	53
13	Appendix A – 3rd Party Components	55
14	Appendix B – <i>WSMO Studio</i> Plug-ins	56
15	Appendix C – Licence	59
16	Appendix D – Changelog	68

List of Figures

3.1	Eclipse plug-in architecture	15
3.2	Eclipse Workbench (courtesy of [Rivieres & Wiegand 04])	16
4.1	Wizard selection dialog	19
4.2	Preference dialog (WSMO editors)	20
4.3	Preference dialog (WSML text editor)	20
4.4	Default <i>WSMO Studio</i> perspectives	21
4.5	Project Navigator	21
5.1	WSMO Navigator – ontology	24
5.2	WSMO Navigator – web service	25
5.3	Non-functional Properties Editor	26
5.4	Ontology Editor	27
5.5	Attribute Editor	28
5.6	Instance Editor	29
5.7	WSML Text Editor	30
6.1	Choreography Editor	33
6.2	State Signature Editor	34
6.3	Rule Viewer	34
6.4	Rule Editor	35
6.5	Rule Editor (composite rules)	35

6.6	Choreography definition in the WSML text editor	36
7.1	Repository Perspective	38
7.2	Repository import – confirmation dialog	40
8.1	WSML validator	42
9.1	Reasoner configuration dialog	44
9.2	WSML-Flight satisfiability check	45
9.3	WSML-Flight satisfiability check	45
10.1	Infrawebs Axiom Editor	47
10.2	WSMO Visualiser	48
10.3	Activating the WSMO Visualiser	49
11.1	Update site – choosing an update site	51
11.2	Update site – choosing a feature to install	51
11.3	Update site – feature licence	52
11.4	Update site – installing a feature	52

List of Tables

1	List of abbreviations	7
13.1	3 rd party components distributed with <i>WSMO Studio</i>	55
14.1	wsmo4j plug-in	56
14.2	Runtime plug-in	56
14.3	WSMO Editor plug-in	57
14.4	Choreography editor	57
14.5	Repository plug-in	57
14.6	ORDI adapter	58
14.7	Web Service adapter	58
14.8	Web Service adapter	58

Acronym	Explanation
API	Application Programming Interface
DTD	Document Type Definition
EPL	Eclipse Public Licence
GUI	Graphical User Interface
IME	Integrated Modelling Environment
J2SE	Java 2 Platform Standard Edition
MVC	Model–View–Controller
NFP	Non-Functional Property
OS	Operating System
OSGi	Open Services Gateway Initiative
OWL	Web Ontology Language
LGPL	GNU Lesser General Public Licence
SWS	Semantic Web Services
SWT	Standard Widget Toolkit
UDDI	Universal Discovery, Description and Integration
WSDL	Web Service Description Language
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
WSMX	Web Service Execution Environment
XML	eXtensible Markup Language

Table 1: List of abbreviations

Acknowledgements

WSMO Studio is partly funded by the European Commission under the IST projects DIP¹ (FP6-507483) and InfraWebs² (FP6-511723).

Barry Norton and Carlos Pedrinaci from the Open University provided valuable feedback for improving *WSMO Studio*.

©2004–2006 [Ontotext Lab.](#) / [Sirma Group](#)

¹<http://dip.semanticweb.org>

²<http://www.infrawebs.org>

Chapter 1

Introduction

The goals of the *WSMO Studio* effort are:

- Providing a GUI tool that assists the users working in the WSMO domain with tasks related to semantic web service annotation.
- Providing a extensible tool and architecture that will allow 3rd parties to integrate and extend WSMO Studio functionality

This guide presents the such an integrated modelling environment, called *WSMO Studio* – a standalone, Eclipse based application that supplies the following functionality:

- Creating WSMO descriptions of ontologies, goals, web services and mediators
- Export and import of the WSMO descriptions (supported languages and formats are WSML, WSML-XML and OWL-DL)
- front-end to service, goal, mediator and ontology repositories (such as IRS-III¹ or WSMX²)
- Creating WSMO centric choreography descriptions

The target audience of this document consists of:

- End users that will use *WSMO Studio* to work with WSMO service descriptions.
- Solution providers that provide new functionality relevant to WSMO/WSMX and that would like to integrate this functionality with *WSMO Studio*.

¹<http://kmi.open.ac.uk/projects/irs/>

²<http://www.wsmx.org>

This document is structured as follows:

- [chapter 2](#) contains instructions for installing *WSMO Studio*
- [chapter 3](#) provides a very brief introduction to Eclipse and its used interface.
- [chapter 4](#) introduces the WSMO Studio user interface that is common across all plug-ins
- [chapter 5](#) presents the WSMO editor plug-in of WSMO Studio, which is used to create the descriptions of all WSML elements: ontologies, services, goals and mediators
- [chapter 6](#) provides details about the Choreography Editor, which is used to create WSMO centric choreography descriptions of the services
- [chapter 7](#) presents the ways to connect to remote ontology / goal / service repositories from *WSMO Studio*
- [chapter 8](#) introduces the WSML Validator which is used to detect problems and inconsistencies within WSML descriptions
- [chapter 9](#) presents the WSML-Flight reasoner that can perform satisfiability checks over the created WSML ontologies
- [chapter 10](#) presents several 3rd party plug-ins that can be used with *WSMO Studio*: the IRS-III adapter, the Infrawebs Axiom Editor and the WSML Visualiser.
- [chapter 11](#) provides details on using the *WSMO Studio* update site to install new feature and update existing features.
- [chapter 13](#) lists all 3rd party libraries incorporated within *WSMO Studio* and their respective licences
- [chapter 14](#) contains short technical summaries about all *WSMO Studio* plug-ins
- [chapter 15](#) presents the LGPL licence, under which *WSMO Studio* is licenced

note:

A lot of additional information (both for end users and for developers) is available at the WSMO Studio web site: <http://www.wsmostudio.org>

Chapter 2

Installing *WSMO Studio*

2.1 Requirements

WSMO Studio requires:

- Java Runtime Environment¹ 1.5+
- Eclipse² 3.1

note:

*If you are using the standalone distribution then you **don't** need to download and install Eclipse separately, since it is included in the distribution)*

2.2 Licence

WSMO Studio is distributed under the GNU Lesser General Public Licence (LGPL), available in [chapter 15](#).

note:

WSMO Studio incorporates 3rd party components distributed under an Apache Software Licence³, GNU Lesser General Public Licence (LGPL)⁴ or an Eclipse Public Licence⁵. Before installing, make sure that you are acquainted with the respective licenses (more information is available in [chapter 13](#)).

¹<http://java.sun.com/j2se/1.5.0/download.jsp>

²<http://www.eclipse.org/downloads/>

³<http://opensource.org/licenses/apache2.0.php>

⁴<http://opensource.org/licenses/lgpl-license.php>

⁵<http://opensource.org/licenses/eclipse-1.0.php>

2.3 Installation Types

WSMO Studio can be used in two modes: as a standalone application (*standalone* mode), or as part of an existing Eclipse deployment (*embedded* mode).

2.3.1 Standalone Mode

To use *WSMO Studio* in a standalone mode, you should only download the latest distribution from <http://www.wsmostudio.org/download.html> and unzip the archive.

2.3.2 Embedded Mode

If you already have an existing Eclipse deployment, then you can only download individual *WSMO Studio* plug-ins from http://sourceforge.net/project/showfiles.php?group_id=119791 and copy them into your *ECLIPSE/plugins* directory. The individual plug-ins are detailly described in [chapter 14](#).

Keep in mind that the following plug-ins are mandatory:

- wsmo4j plug-in
- WSMO Studio Runtime plug-in
- WSMO plug-in

Check out [chapter 14](#) for more details on individual plug-ins.

Chapter 3

Introduction to Eclipse

This section will provide a brief overview of the Eclipse platform¹, so that concepts and terms that will be used throughout the Guide are properly explained. A more detailed overview of Eclipse is available in [Rivieres & Wiegand 04].

3.1 Plug-in Based Architecture

There are two components in the Eclipse architecture – a small runtime kernel and a set of plug-ins. Plug-ins represent the smallest unit of functionality within Eclipse and the runtime kernel is responsible for the plug-in lifecycle management.

3.1.1 Runtime Kernel

The Eclipse runtime is a small, OSGi microkernel. The OSGi Service Platform [OSGi Alliance 04] presents a standardised environment for managing the lifecycle of software components. With the adoption of OSGi in Eclipse v3.0, the Eclipse Platform now presents a dynamic plug-in model (e.g. plug-ins can be added, removed or updated at runtime, without restarting the Eclipse system). When the Eclipse system (e.g. an Eclipse based application) is started, the microkernel discovers the plug-ins deployed within the system and builds a dynamic registry of the available plug-ins.

¹<http://www.eclipse.org>

3.1.2 Plug-ins

Plug-ins encapsulate the smallest unit of functionality within Eclipse. Apart from the microkernel, everything else in Eclipse is a plug-in (including the GUI, the tools, etc.). In other words, with the exception of the microkernel, which is responsible only for plug-in management, all the functionality provided to the end user is in the form of plug-ins (usually a small functional feature can be delivered as a single plug-in, but complex functionality is delivered in the form of a set of contributing plug-ins).

A plug-in is usually comprised of:

- A plug-in descriptor
- The core plug-in functionality in the form of one or more Java library archives
- 3rd party Java libraries used by the plug-in
- Other resources required by the plug-in (images, read-only files, etc.)

Plug-ins may depend on other plug-ins, extend other plug-ins, be bundled together with other plug-ins or be composed from a set of fragments.

Plug-in Descriptor

The plug-in descriptor (or *manifest*) is an XML file – called *plugin.xml* – which presents the declarative description of the plug-in. The descriptor contains information such as the plug-in name, ID, version, dependencies on other plug-ins, exposed functionality, extension points and extensions. The plug-in descriptor describes how the plug-in will be integrated in the runtime environment.

Dependencies

A plug-in may reuse functionality from other plug-ins in the system. Such a dependency is declared in the plug-in descriptor so that the Eclipse runtime may ensure that all required (prerequisite) plug-ins are available before activating a particular (dependent) plug-in.

Extension Points

A plug-in may allow other plug-ins to extend its functionality (the former is usually called a *host plug-in* and the latter are *extender plug-ins*). The host plug-in declares one or more *extension points* in its plug-in descriptor, while extender plug-ins specify a set of *extensions* in their descriptors (e.g. which extension points of a host plug-in are being extended).

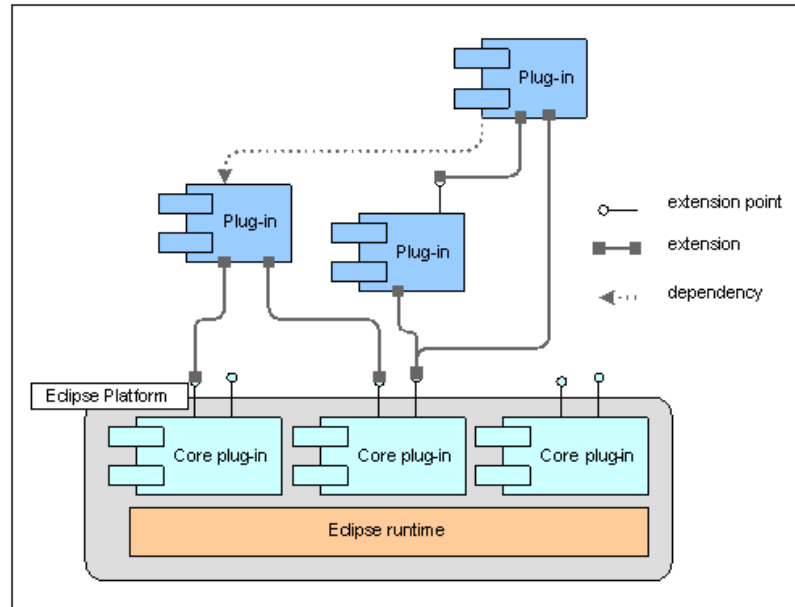


Figure 3.1: Eclipse plug-in architecture

Figure 3.1 presents the Eclipse plug-in architecture and relationships between individual plug-ins (extension or dependency).

The extension mechanism in Eclipse allows for loose coupling between components (the host plug-ins are unaware of the eventual extenders, apart from the fact that extenders should adhere to the extension point specification). In addition, the declarative dependency specification improves the modularisation of the platform.

Plug-in Activation

When the Eclipse platform is started the Eclipse runtime (microkernel) inspects the set of plug-ins deployed within the system and builds a plug-in registry. At this point plug-ins are still not activated. A plug-in is activated only when its functionality is required (implicitly by other plug-ins or explicitly by the user). With such a load-on-demand model the Eclipse platform offers optimal resource consumption, even if there are thousands of plug-in deployed within the system.

3.2 User Interface

The main Eclipse UI concepts are:

- *Views* and *editors* – containers for logically and functionally related UI sub-components (text areas, buttons, etc.).
- *Perspectives* – containers for logically and functionally related views and editors. Only one perspective is visible at any moment but several perspectives may be active. A perspective specifies the layout and visibility of the enclosed views and editors. It is recommended that a perspective provides all the functionality related to a specific task, so that the user won't have to switch between different perspectives to accomplish his task.

Figure 3.2 presents a screenshot from the Workbench.

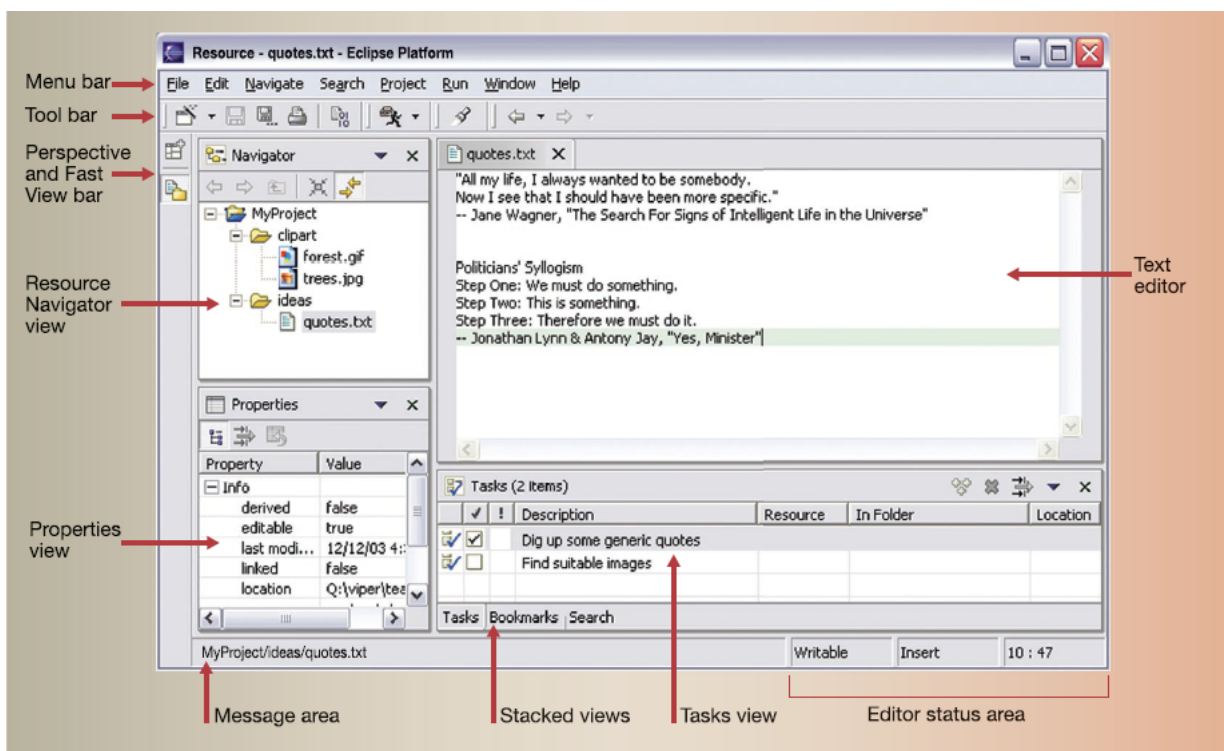


Figure 3.2: Eclipse Workbench (courtesy of [Rivieres & Wiegand 04])

Chapter 4

WSMO Studio UI

This chapter introduces the common elements of the *WSMO Studio* user interface. Note that when plug-ins are added to, or removed from *WSMO Studio* the interface may change as well.

Chapters [chapter 5](#), [chapter 6](#) and [chapter 7](#) will provide detailed information on the *WSMO Studio* user interface.

4.1 Menus

The standalone *WSMO Studio* application menu consists of the following items:

- *File*
 - *New* – starts a Wizard for creating elements such as (see [Figure 4.1](#)):
 - * WSMO Projects
 - * Ontologies
 - * Goals
 - * Web Services
 - * Mediators
 - *Save / Save All* – saves the modified WSML descriptions
 - *Import* – import files into the active workspace. This menu also activates the OWL-DL, RDF and XML-WSML import wizards, that allow import into WSML from OWL-DL, RDF and XML representation of WSML respectively¹.

¹see [[deBruijn et al. 05](#)] for details on the WSML mapping to XML, RDF, OWL-DL

- *Export* – export files from the active workspace. This menu also activates the XML-WSML export wizard, that allows exporting of WSML files into the XML representation of WSML (Note that export to RDF and OWL-DL is not supported at present).
- *Exit* – exits the application.
- *Edit* – standard editing functionality (copy & paste, undo & redo, etc.)
- *Project* – open / close projects
- *Window* – standard Eclipse settings for perspectives, views and preferences (see also section 4.2)
- *Help*
 - *Help Contents* – WSMO Studio at present does not provide integrated help contents.
 - *Software Updates* – the update site functionality. *WSMO Studio* provides an option to automatically get updates of the installed plug-ins from the *WSMO Studio* web site. See chapter 11 for details.

4.2 Wizards

The following wizards are available by default in *WSMO Studio* (see Figure 4.1):

- WSMO Project wizard
- Ontology wizard
- Goal wizard
- Web Service wizard
- Mediator wizard

Wizards at present do not provide a step-by-step assistance for creating WSMO elements but simply activate the respective set of views and editors that provide the functionality for specifying WSMO elements.

4.3 Preferences

WSMO Studio defines a set of preference, that allow workbench customisations to be preserved between working sessions (see Figure 4.2 and Figure 4.3). The set of preferences may vary depending on the installed plug-ins.

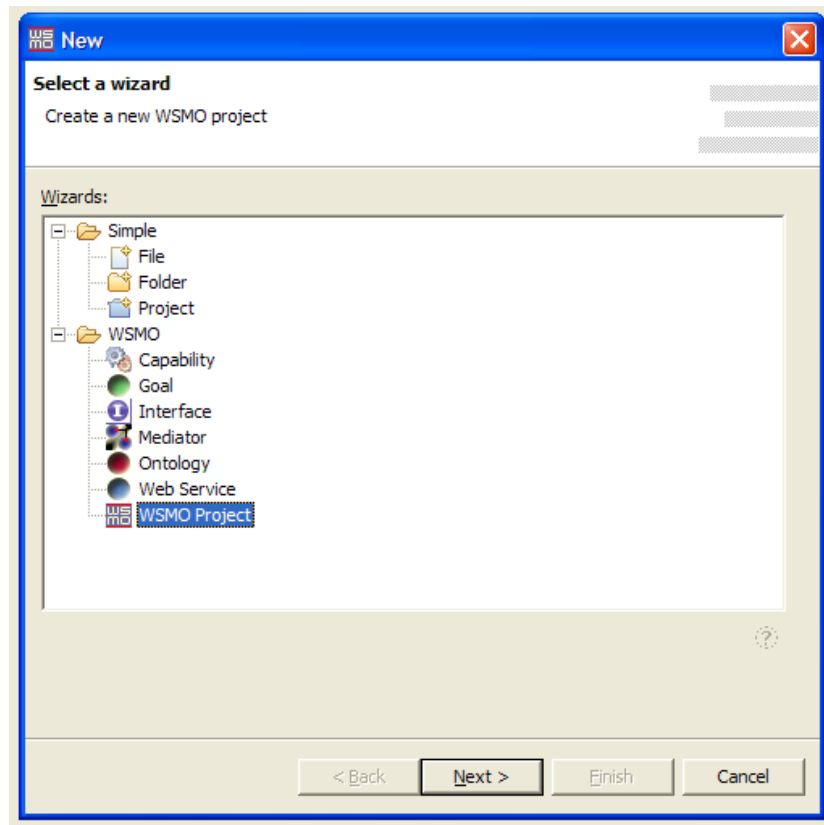


Figure 4.1: Wizard selection dialog

4.4 Perspectives

WSMO Studio defines two perspectives at present:

- WSMO perspective
- Repository perspective

The WSMO Perspective groups together functionality related to creating descriptions of WSMO elements (ontologies, goals, services and mediators) and exporting these in WSML formats. Most of the *WSMO Studio* plug-ins contribute to this perspective.

The Repository perspective groups together functionality for accessing remote ontology, goal, service and mediator repositories. The Repository plug-in and its various adapters contribute to this perspective.

note: *Whether a perspective is active (within a particular Eclipse deployment) depends only on the user's preferences. By default, the WSMO and Repository perspectives are active in the standalone version of WSMO Studio (e.g. the toolbar should look like the one shown in*

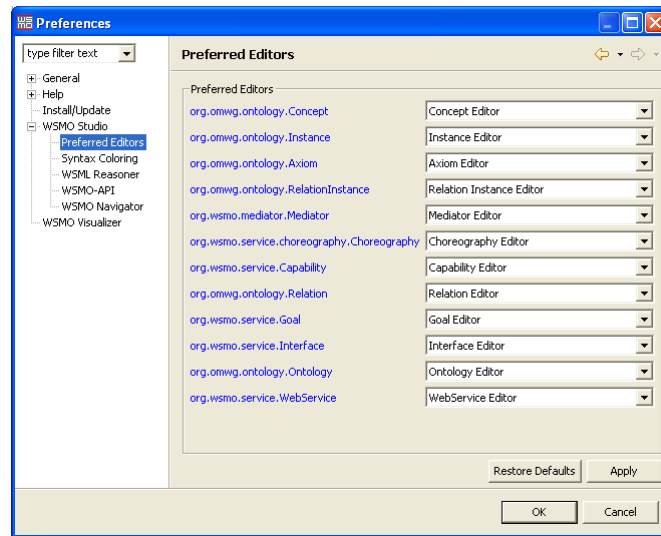


Figure 4.2: Preference dialog (WSMO editors)

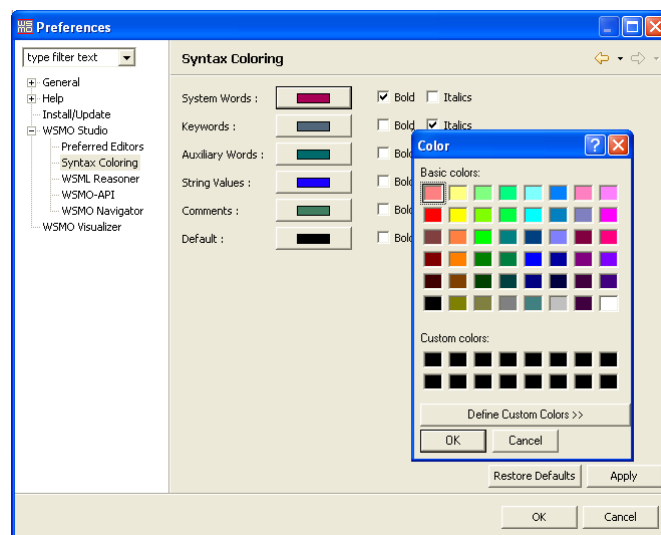


Figure 4.3: Preference dialog (WSML text editor)

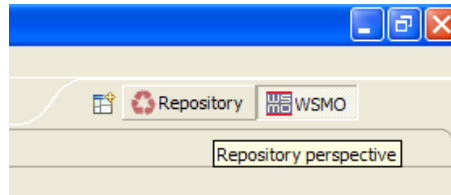


Figure 4.4: Default *WSMO Studio* perspectives

Figure 4.4), but if some perspective is not active, it can always be activated from the menu (*Window* → *Open perspective* → *Other*)

4.5 Project Navigator

The Project Navigator (see [Figure 4.5](#)) presents a tree view of the projects in the current workspace, together with the WSMML files associated with a project. Double clicking a WSMML file from a project will activate the respective views and editors associated with the type of WSMO entity (Ontology, Web Service, Mediator, Goal).

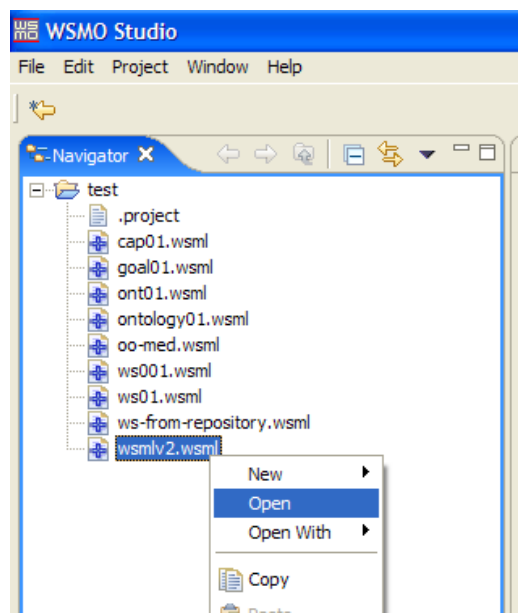


Figure 4.5: Project Navigator

Documents can also be activated from the context menu (see [Figure 4.5](#)). At present there are three possible way of visualising and editing a WSMML file:

- the default form based editors and views ([chapter 5](#) and [chapter 6](#))

- the text editor ([section 5.3](#))
- the WSML Visualiser (adapted from WSMT) which provides a hyperbolic tree based interface ([section 10.3](#))

Chapter 5

The WSMO Editor

When creating descriptions of WSMO elements (such as ontologies, services, goals and mediators), the user works in the *WSMO perspective*, or the WSMO Editor which will be presented in this chapter.

The WSMO perspective consists of the following views and editors:

- WSMO navigator, located in the bottom-left part (see [Figure 5.1](#))
- Editors that occupy most of the right part of the UI area
- Properties tab that is located in the bottom-right part of the UI area

The following sections contain more details on the intended usage of these parts of the UI.

5.1 WSMO Navigator

The WSMO Navigator (see [Figure 5.1](#) and [Figure 5.2](#)) presents summary view that is specific for the selected WSMO entity (ontology, service, goal or mediator).

If the entity is an ontology then the concept / relation hierarchy is shown as well as the list of axiom definitions present in the ontology.

The context menu is element specific as well. The context menu actions, depending on the type of element that is selected, are:

- Concept (Relation) – add sub-concept (relation), add instance, edit concept (relation), remove concept (relation)

- Instance – edit instance, remove instance
- Axiom – edit axiom, remove axiom

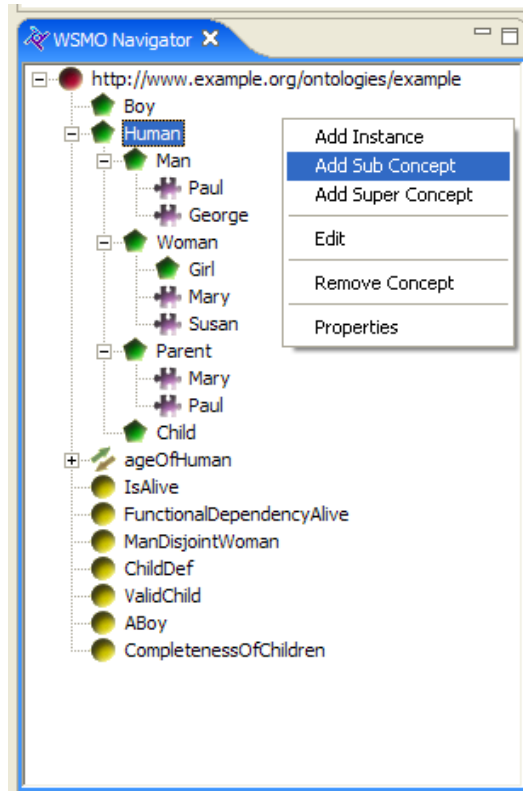


Figure 5.1: WSMO Navigator – ontology

If the WSMO entity is a Web Service then WSMO Navigator shows the service capability and the list of interfaces associated with the Web Service (see [Figure 5.2](#)). The context menu is again element specific.

In a similar manner, the WSMO Navigator will present specific information for the other WSMO elements: Mediators and Goals.

5.2 Editors

When a WSMO entity is selected (e.g. the respective WSML file is opened), the entity specific views and editors are activated on the right hand side of the UI area. There are a number of UI elements that are common for all entities as well as several entity specific UI elements.

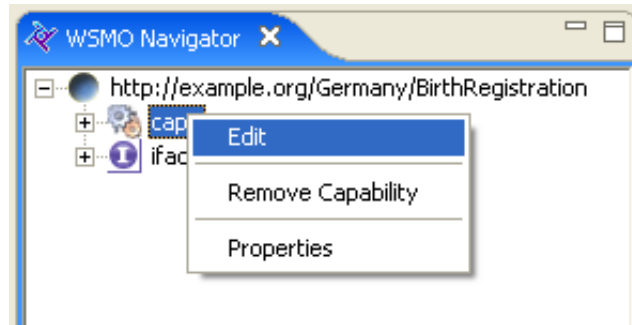


Figure 5.2: WSMO Navigator – web service

5.2.1 Common Editors

The common UI elements present the information that is common for all WSMO entities, such as:

- Identifier¹
- Non-functional properties (see [Figure 5.3](#))
- Used namespaces
- Imported ontologies
- Used mediators

5.2.2 Ontology Editors

When an ontology is edited, all the common UI components (views and editors for identifier, non-functional properties (NFP), used namespaces, imported ontologies and mediators) are activated (see [Figure 5.4](#)).

More specific editors are available that provide means to specify individual ontology elements:

- Concept editors
- Relation editors
- Instance editors
- Axiom editors

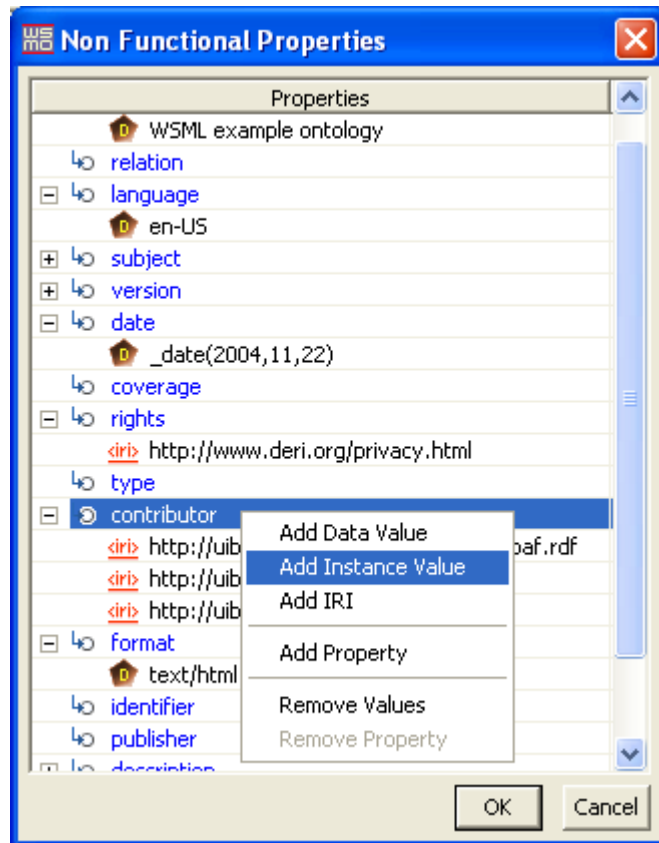


Figure 5.3: Non-functional Properties Editor

These specific editors are activated when an element from the WSMO Navigator ([section 5.1](#)) is chosen for editing, either by double clicking on the respective element or by choosing "Edit?" from the context menu.

5.2.3 Concept / Relation Editor

The concept editor provides means for specifying an ontology concept and its:

- Attributes: add / remove attributes (see [Figure 5.5](#))
- Super concepts: add / remove super concepts
- Instances: add / remove instances (see [Figure 5.6](#))

¹Note that identifiers are *immutable*, e.g. once specified they cannot be modified.

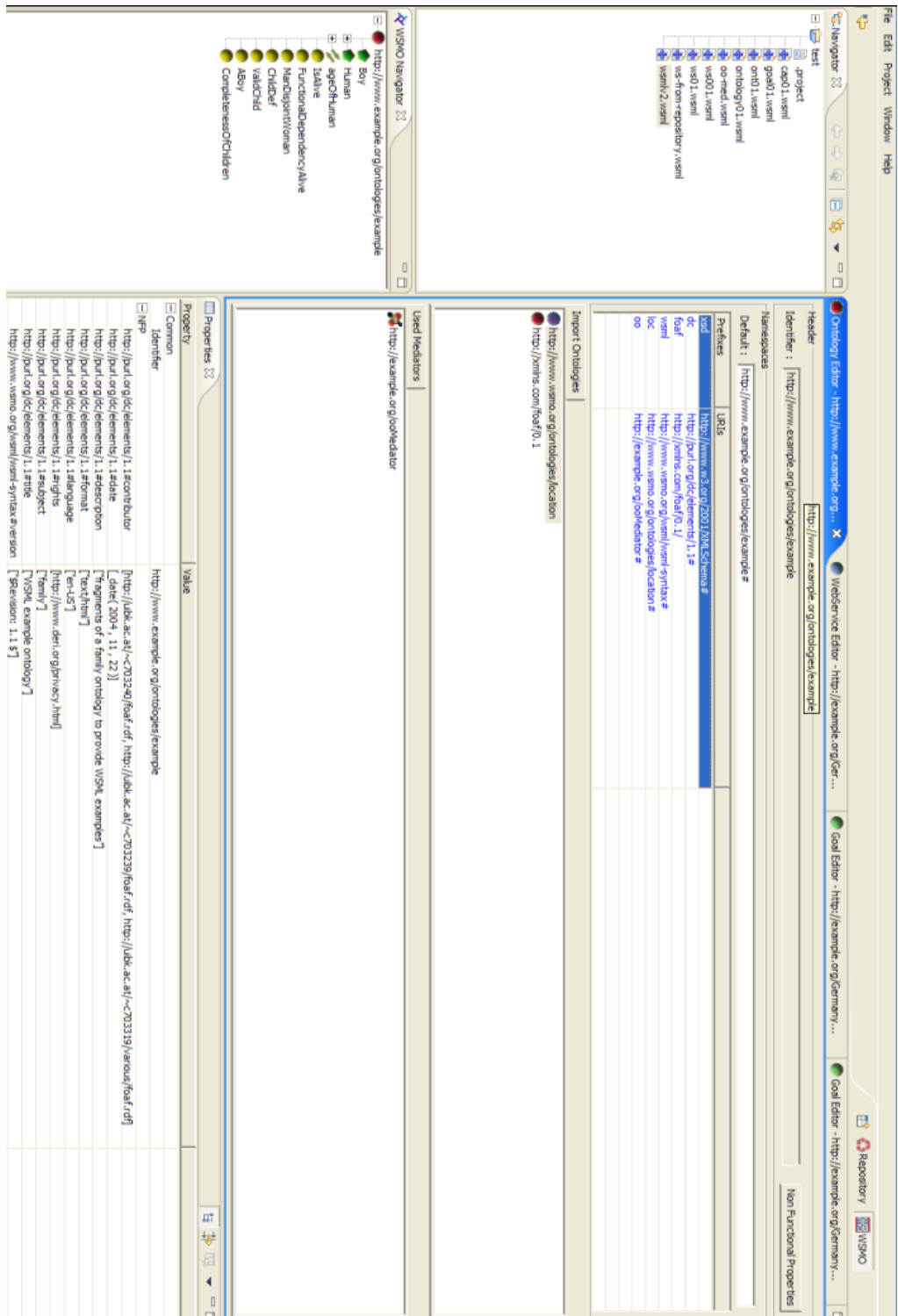


Figure 5.4: Ontology Editor

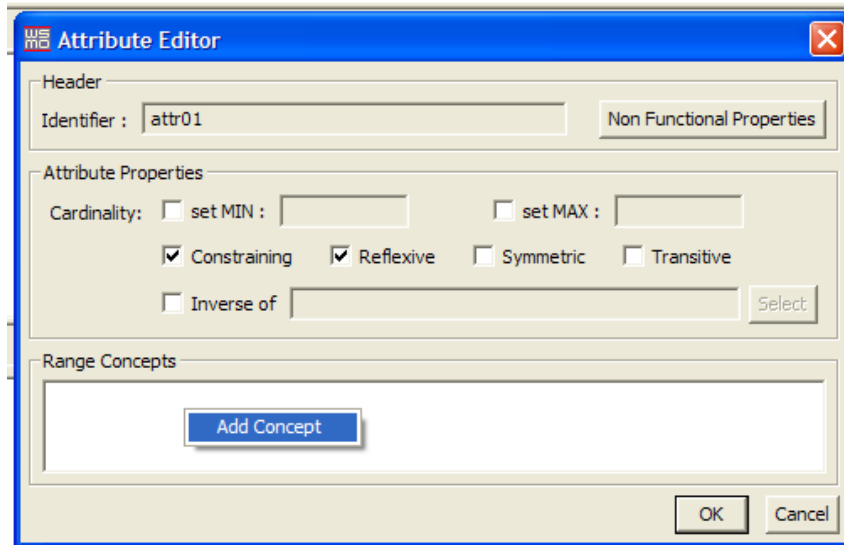


Figure 5.5: Attribute Editor

5.2.4 Instance Editor

The instance editor (Figure 5.6) provides the following functionality:

- Specify the concepts that the instance belongs to
- Specify attribute values for the attributes defined for the concept

5.2.5 Axiom Editor

The Axiom Editor provides means for specifying the logical expressions that define an axiom.

At present, the default logical expression editor is a plain text editor, but more advanced logical expression editors may be seamlessly integrated as well, since the logical expression editor is defined as an extension point. In chapter 10 we will present an extended axiom editor based on the Eclipse Graphical Editing Framework².

5.2.6 Goal / Web Service Editors

The Goal / Web Service editors provide means for specifying the capability and the list of interfaces associated with a specific Goal / Web Service.

²<http://www.eclipse.org/gef/>

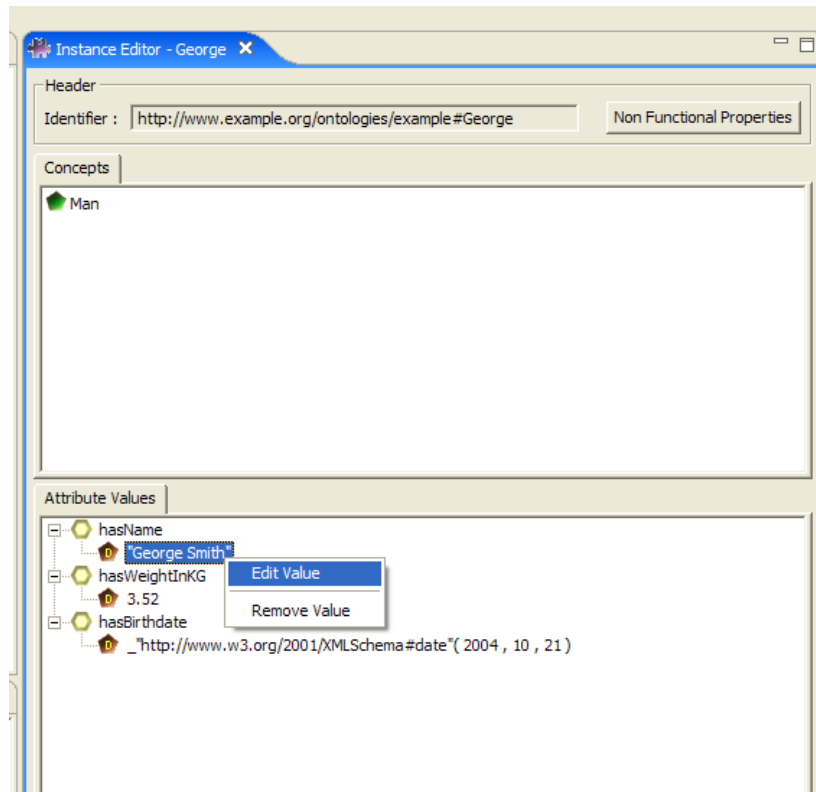


Figure 5.6: Instance Editor

The Capability Editor is used to specify the pre-conditions, post-conditions, assumptions and effects comprising the Capability.

The Interface Editor is used to specify the WSMO choreography and specifications ([Scicluna *et al.* 05]) for the service. The Choreography Editor will be presented in details in chapter 6.

5.2.7 Mediator Editors

Mediator Editors provide means to specify the WSMO description of a mediator:

- source and target components
- the mediation service URI.

Note that the mediator editor does not provide any functionality for implementing the actual mediation service (such as mapping rules for ontology mediation or actual code that will

perform the mediation). The mediator service should be implemented according to the specific Semantic Web Service execution environment (for example IRS-III³ or WSMX⁴).

Check out [Cabral & Domingue 05] for details on mediator services in IRS-III.

5.3 WSML Text Editor

The WSML text editor allows editing of the plain text WSML descriptions (see Figure 5.7).

```
relation ageOfHuman (ofType Human, ofType _integer)
  nfp
  dc#relation hasValue {FunctionalDependencyAge}
endnfp

axiom IsAlive
  definedBy
    ?x[isAlive hasValue _boolean("true")] :-
    naf ?x[hasObit hasValue ?obit] memberOf Human.
    ?x[isAlive hasValue _boolean("false")]
    impliedBy
    ?x[hasObit hasValue ?obit] memberOf Human.
```

Figure 5.7: WSML Text Editor

The editor supports syntax highlighting and the list of the predefined WSML keywords could be additionally extended by the user. The highlighting colours are customisable by the end user as well (see Figure 4.3).

³<http://kmi.open.ac.uk/projects/irs/>

⁴<http://www.wsmx.org>

Chapter 6

Choreography Editor

The Choreography Editor provides functionality for creating WSMO based choreographies based on [Scicluna *et al.* 05].

6.1 WSMO Based Choreography

A WSMO based choreography is comprised of:

- A *state signature* that specifies the state ontology for the choreography. The concepts and relations of this ontology will be used to express the state of the choreography, e.g. the actual states are instances of concepts / relations from the state ontology. In addition to the specification of the exact state ontology, the state signature also specifies several *roles* (also called *modes*) for the concepts and relations of the state ontology. There are five predefined roles:
 - *Static* (default) – the instances of such concepts/relations cannot be changed by the choreography execution
 - *In* – instances can only be read by the choreography, but can be changed by the environment (e.g. the agents external to the service). A *grounding* may be provided, that specifies means for the environment to modify such instances.
 - *Controlled* – instances of such concepts / relations can be created and modified only by the choreography execution and cannot be accessed by the environment
 - *Shared* – instances can be created or modified both by the choreography execution and by its environment. A grounding mechanism is specified as well.
 - *Out* – instances can be created / modified by the choreography execution, but the environment can only read such instances. A grounding mechanism must be specified, that provides read-only access for the environment.

- A set of *transition rules* that express state changes (e.g. changes in the set of instances that comprise the choreography state). The most basic form of transition rule are update functions, which add / remove / modify instances of concepts and relations. The three different types of update functions are:

- *add*(*fact*)
- *delete*(*fact*)
- *update*(*fact_{old}* → *fact_{new}*)

... where a fact is either a membership fact (e.g. "X memberOf Y") or an attribute fact (e.g. "X hasValue Y"). With the help of basic update rules, more complex transition rules can be defined recursively:

- **if** *condition* **then** *rules* **endif**
- **forAll** *variables* **with** *condition* **do** *rules* **endForAll**
- **choose** *variables* **with** *condition* **do** *rules* **endChoose**

... where conditions are restricted WSML logical expressions and variables are WSML variables (see [deBruijn *et al.* 05] for details).

6.2 User Interface

When a Web Service is selected in the Project Navigator (top-left pane of the user interface), the WSMO Navigator, located in the bottom-left part of the user interface, will show a summary of the Web Service components (see [Figure 6.1](#)):

- the Capability of the service
- the Interface of the service
- the Choreography, part of the interface definition

When the choreography is open for editing (by double-clicking on the respective element or selecting Edit from the context menu), the choreography editor is activated.

The interface area of the Choreography Editor is divided in two parts, for the State Signature editor and the Rules editor.

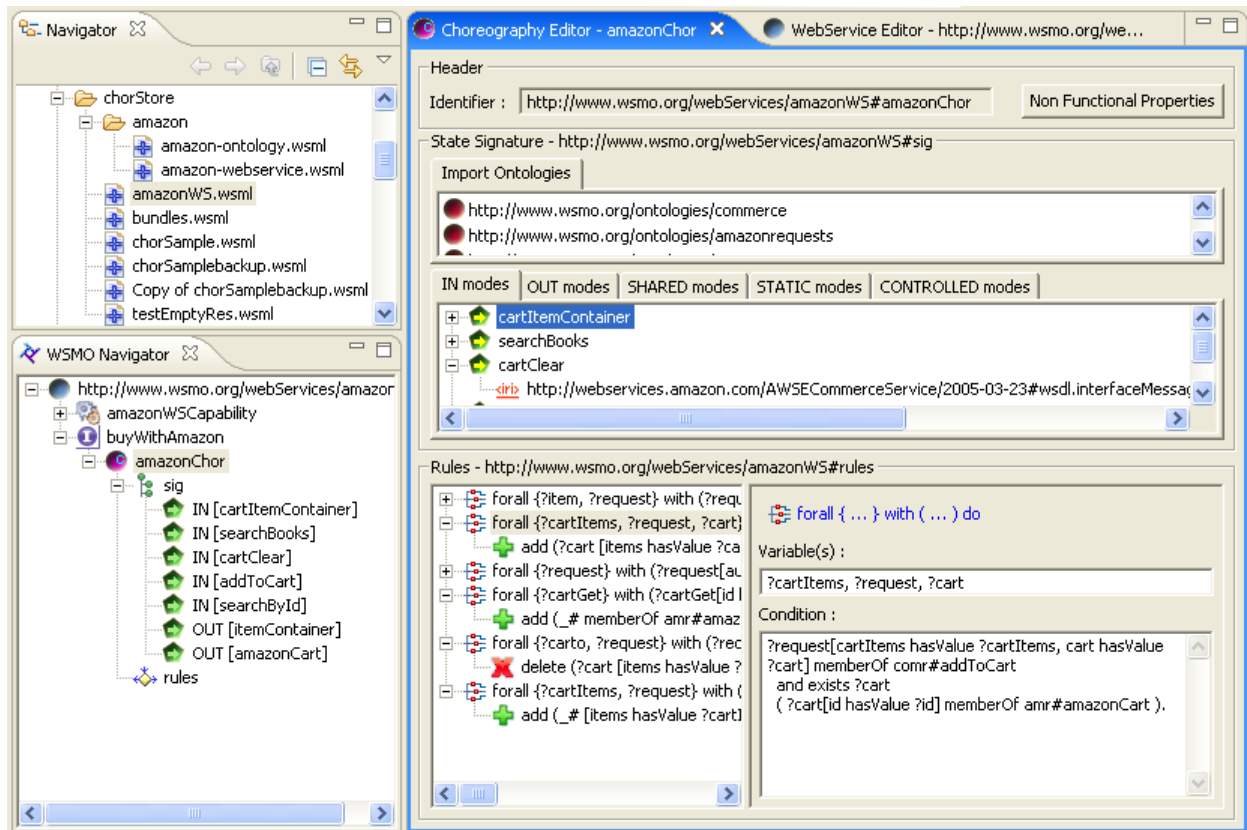


Figure 6.1: Choreography Editor

6.2.1 State Signature Editor

The State Signature editor provides functionality for describing the state signature of the choreography, e.g. importing a state ontology, and defining the modes / roles of the concepts and relations that will define the instance states (see Figure 6.2):

For each type of mode, the following functionality is provided:

- Adding a new mode (entity) – instances of the selected concept / relation will comprise the state of the choreography
- Removing a mode (entity)
- Add / remove grounding (only for in, out and shared modes) – provides a grounding mechanism for the selected mode (for example, a reference to a WSDL message)

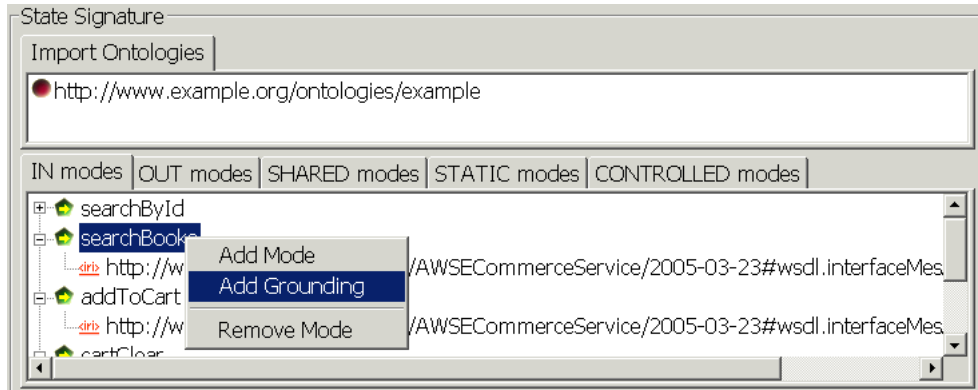


Figure 6.2: State Signature Editor

6.2.2 Transition Rules Editor

The Rules component provides functionality for visualising and editing transition rules. The left part of the component is comprised of a tree-based viewer that shows the list of transition rules, defined for the choreography (see Figure 6.3). If the rules are composite (e.g. *if-then*, *forAll* and *choose* rules), then the nested rules will be shown in the hierarchy too.

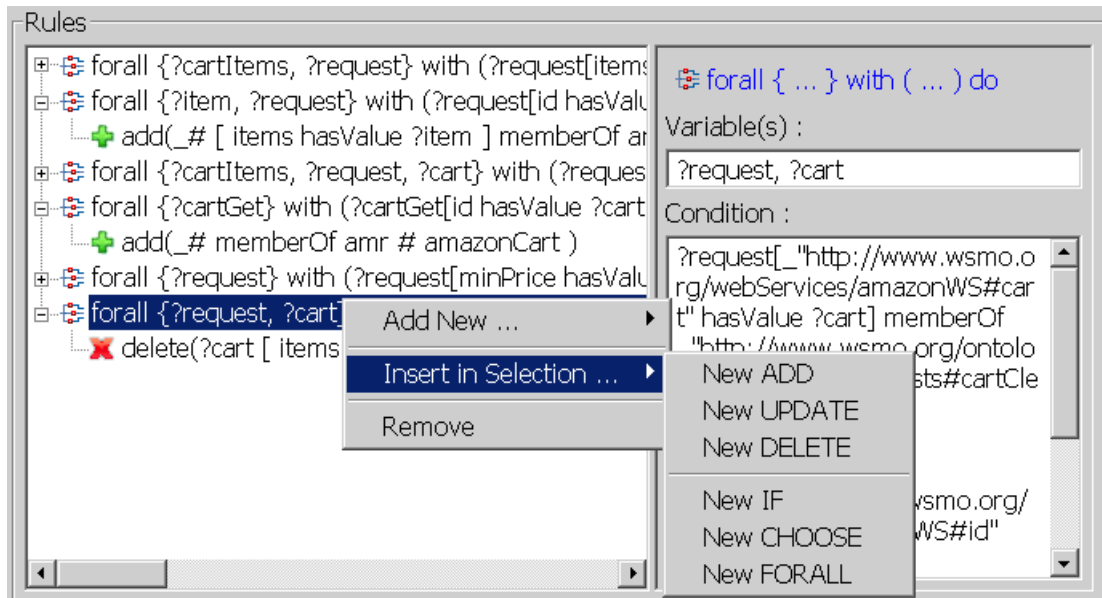


Figure 6.3: Rule Viewer

New rules can be added¹, or existing rules can be removed from the hierarchy by using the actions in the context menu.

¹Note that the transition rules are *unordered*, e.g. the specific position in which they appear in the list is insignificant.

When a particular rule is selected, a rule editor is activated in the bottom-right part of the user interface (see [Figure 6.4](#) and [Figure 6.5](#)). The specific user interface components of the editor differ according to the nature of the rule – the editor for simple rules ([Figure 6.4](#)) is comprised of a single text area for providing the contents of the rule.

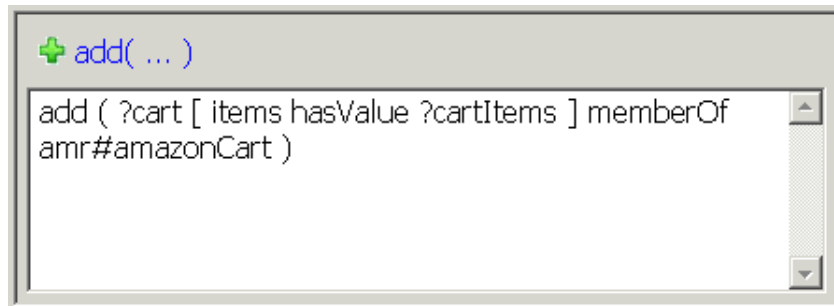


Figure 6.4: Rule Editor

If the rule is a composite one, then the editor also provides means for specifying conditions and variables ([Figure 6.5](#)). The conditions (in the form of restricted WSML axioms) are validated upon saving the choreography definition using the validator described [chapter 8](#).

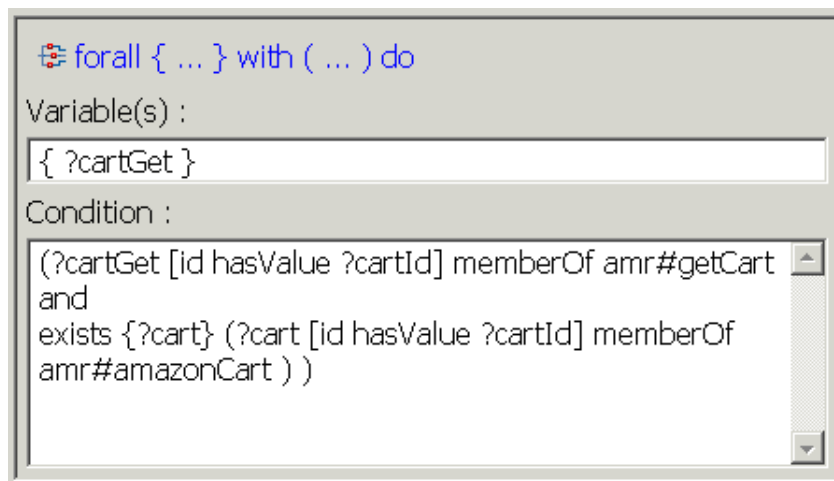


Figure 6.5: Rule Editor (composite rules)

The complete choreography definition can also be viewed and edited in the default WSML text editor, described in [section 5.3](#) (see [Figure 6.6](#)):

```
amazonWS.wsml x
interface buyWithAmazon
  choreography amazonChor
  stateSignature sig
  importsOntology(
    _ "http://www.wsmo.org/ontologies/commerce",
    _ "http://www.wsmo.org/ontologies/amazonrequests",
    _ "http://www.wsmo.org/ontologies/commercerequests"
  )

  in
    amr#searchBooks withGrounding _ "http://webservices.amazon.com/AWSECommerceService/2005-03-23#ws
    amr#searchById withGrounding _ "http://webservices.amazon.com/AWSECommerceService/2005-03-23#wsd

  out
    commerce#itemContainer withGrounding (
      _ "http://webservices.amazon.com/AWSECommerceService/2005-03-23#wsdl.interfaceMessageReferen
      _ "http://webservices.amazon.com/AWSECommerceService/2005-03-23#wsdl.interfaceMessageReferen
    )

  transitionRules rules
  /*
  * Search by keywords, author, title, price range and create a list of items. The
  * condition checks also that such an item with the specified attribute values
  * exists in the state.
  */
  forall {?request} with
    (?request[
      keywords hasValue ?keywords,
      author hasValue ?author,
      title hasValue ?title,
      minPrice hasValue ?minPrice,
      maxPrice hasValue ?maxPrice
    ] memberOf amr#searchBooks and
    exists (?book) (?book[
      keywords hasValue ?keywords,
      author hasValue ?author,
      title hasValue ?title,
      minPrice hasValue ?minPrice,
      maxPrice hasValue ?maxPrice
    ] memberOf bk#amazonBook
    )) do
    add( # memberOf commerce#itemContainer)
  endForall
```

Figure 6.6: Choreography definition in the WSML text editor

Chapter 7

Working with Repositories

7.1 Repository Perspective

The *Repository perspective* in *WSMO Studio* (Figure 7.1) provides functionality for accessing remote ontology, goal, service and mediator repositories¹.

The UI area of the Repository perspective contains several components:

- Repository Explorer
- Ontologies list
- Mediators list
- Web Services list
- Goals list

7.1.1 Repository Explorer

The Repository Explorer lists the remote repositories that *WSMO Studio* can interact with. When the user connects to a repository (*Open* from the context menu), he will be prompted for the repository specific connection information (for example: Web Service endpoint location, user / password, etc.).

Once *WSMO Studio* is able to connect to the remote repository, the list of ontologies, mediators, services and goals will be populated in the respective UI components.

¹Note that in order to be accessible from *WSMO Studio*, the remote repository should provide a facade interface that the Studio can interact with. More details are available in the Developers Guide

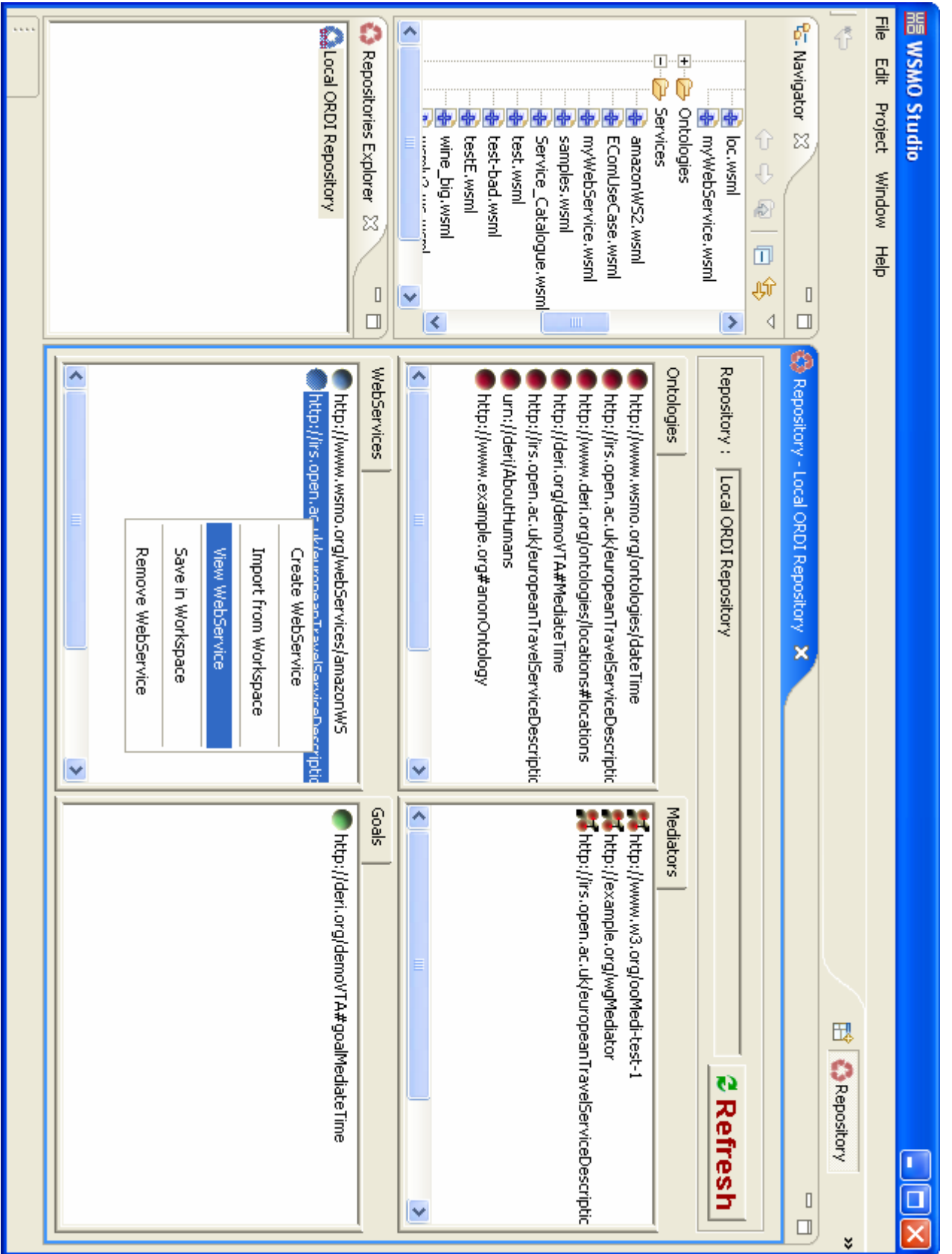


Figure 7.1: Repository Perspective

7.1.2 Repository Views

There are four repository views – for ontologies, mediators, goals and web services. Note that whether all of these WSMO entities are available in a repository depends on the type of repository. Some repositories may publish only ontologies or web services for example.

The context menu of each view presents the following functionality:

- Copying WSMO entities (ontologies, mediators, services and goals) from the workspace (more specifically the Project Navigator described in [section 4.5](#)) into the remote repository
- Copying entities from the remote repository into the workspace (i.e. the reverse of the above operation)
- Removing elements from the remote repository

7.1.3 Import and Export

The sequence of steps that the user should follow in order to be able to copy WSML descriptions from the local workspace into a remote repository is:

1. Create a WSML description using the WSMO Editor (see [chapter 5](#)) and save it in a project in the workspace
2. Switch to the repository perspective (see [Figure 4.4](#))
3. Connect to a remote repository
4. Copy the WSML file into the remote repository (*Import from Workspace* from the context menu)

Note that WSML descriptions stored in a remote repository *cannot* be edited directly. The proper sequence of steps for editing a WSML description from a remote repository is:

1. Copy the WSML description from the remote repository into the local workspace (*Save in Workspace* from the context menu)
2. Switch back to the WSMO perspective (see [Figure 4.4](#))
3. Apply the desired modifications to the WSML description (e.g. edit the WSMO entity using the respective editors from the WSMO perspective)

4. Copy the modified WSMML description from the local workspace back into the remote repository (*Import from Workspace* from the context menu. At this point, the user will be prompted for confirmation whether to overwrite the WSMML description that already exist in the repository (see [Figure 7.2](#))

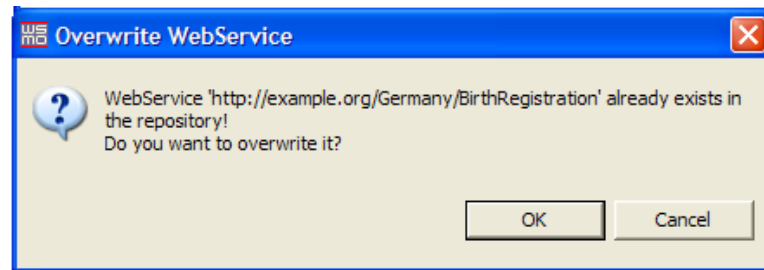


Figure 7.2: Repository import – confirmation dialog

7.2 Integrated ORDI Repository

A local repository, based on the ORDI framework², is available by default in *WSMO Studio*. To use the local repository the user must choose "ORDI Repository" from the context menu of the Repository plug-in.

Another repository adapter, that makes it possible for *WSMO Studio* to interact with the IRS-III server³ is described in [section 10.1](#)

²<http://www.ontotext.com/ordi/>

³<http://kmi.open.ac.uk/projects/irs/>

Chapter 8

WSML Validator

WSMO Studio contains an integrated WSML validator from the `wsmo4j` framework¹.

All errors, warnings and notifications produced from the validator are reflected in the studio's graphical environment (see [Figure 8.1](#)) in the standard *Problems* view of Eclipse. The information associated with each problem is:

- severity,
- explanation message
- problematic location.

If the error concerns parsing problems, the line/position in the text content is indicated.

To avoid unnecessary overhead, validation is performed only when a WSML file is opened or saved. The information about the problems identified is preserved between working sessions. The WSML validation can be disabled from the *WSMO Studio's* preference pages.

¹<http://wsmo4j.sourceforge.net>

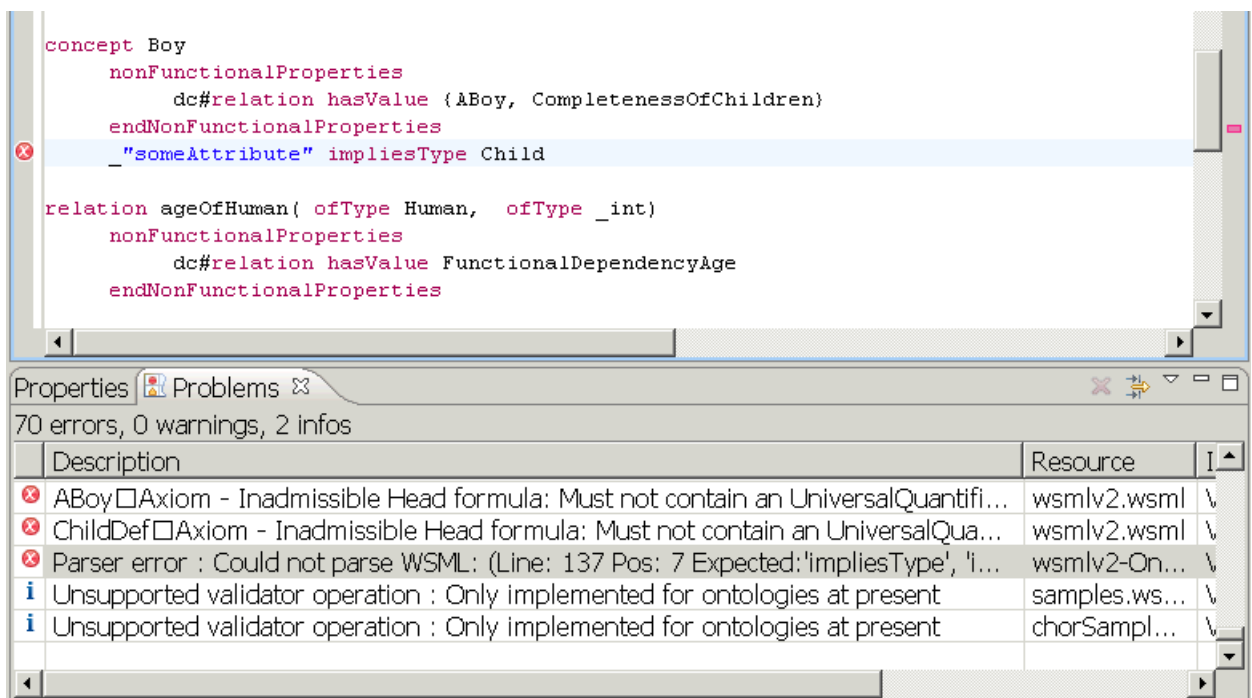


Figure 8.1: WSMML validator

Chapter 9

WSML Reasoner

9.1 Installation

The reasoning related functionality in *WSMO Studio* is comprised of three components:

- the WSML Reasoner GUI (part of *WSMO Studio*, LGPL licence)
- the WSML 2 Reasoning Framework (distributed with *WSMO Studio*, LGPL licence)
- the individual reasoner modules such as *MINS*¹ (GPL licence) or *KAON2*² (custom licence)

note:

Due to the licensing restrictions of the two reasoner modules (MINS and KAON2), they cannot be distributed together with an LGPL distribution, such as WSMO Studio, and the user should download them separately.

In order to configure the reasoners for use within *WSMO Studio* one should follow the steps:

1. The WSML Reasoner GUI and the WSML Reasoning Framework will be installed by default with *WSMO Studio*
2. Download the reasoner module that you prefer (MINS or KAON2). Note that the reasoners provide different functionality and different licences
3. Copy the reasoner jars (*mins-20060202.jar* for MINS, or *kaon2.jar* for KAON2) into the *WSMO_Studio_X.Y.Z/plugins/org.wsmstudio.reasoner.wsml_flight_A.B.C/lib* folder, where *X.Y.Z* is the respective *WSMO Studio* version and *A.B.C* is the version of the reasoner plug-in part of the Studio (check out [Table 14.8](#) for details).

¹<http://dev1.der1.at/mins/>

²<http://kaon2.semanticweb.org/>

9.2 Usage

After the installation steps have been performed, the WSML-Flight reasoner can be used with *WSMO Studio*:

1. Start *WSMO Studio* and configure the reasoner preferences (*Window* → *Preferences* → *WSMO Studio* → *WSML Reasoner*) as shown on [Figure 9.1](#)
2. Try a satisfiability check on some of the sample ontologies (from the context menu choose "Test WSML-Flight satisfiability", see [Figure 9.2](#)). Instead of getting a message about missing reasoners, you should get a message from the WSML Flight reasoner that states whether the ontology is satisfiable or not (see [Figure 9.3](#))

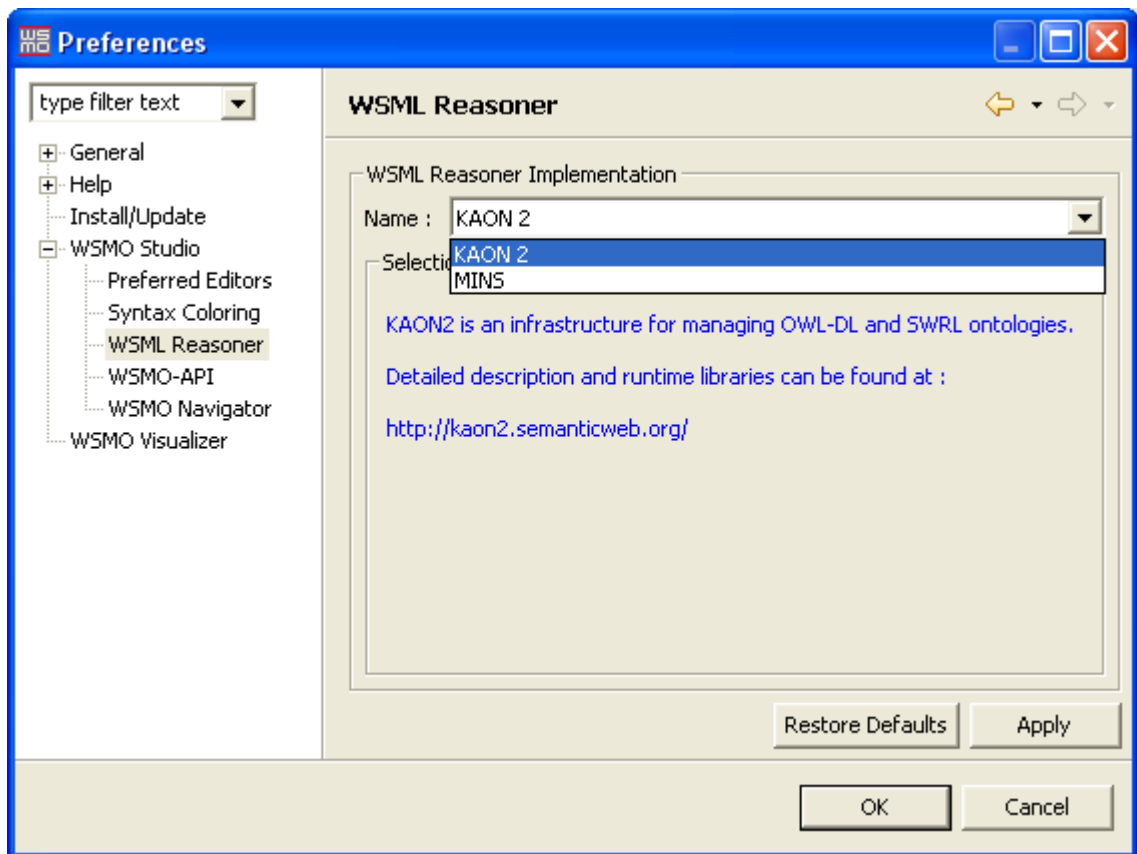


Figure 9.1: Reasoner configuration dialog

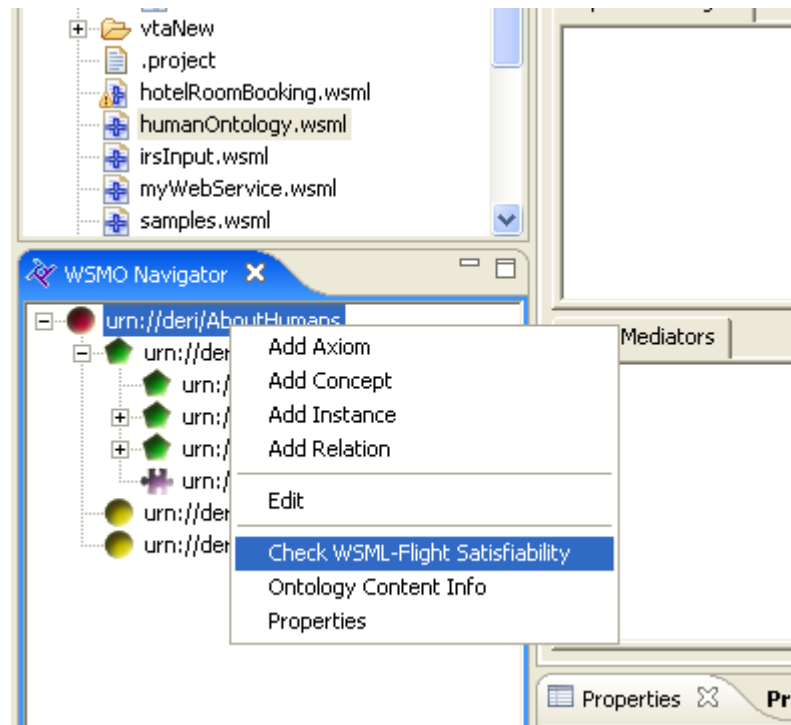


Figure 9.2: WSML-Flight satisfiability check

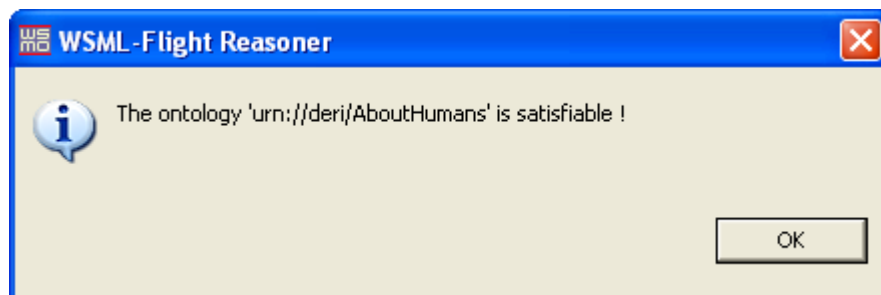


Figure 9.3: WSML-Flight satisfiability check

Chapter 10

3rd Party Plug-ins

10.1 IRS-III Adapter

The IRS-III plug-in¹ is a front-end for the IRS-III server² into *WSMO Studio*. The plug-in supports importing and exporting WSMO entities, such as Goals, Mediators, Web Services and Ontologies, to and from the reasoning server. Furthermore, it allows users to achieve Goals using the IRS-III directly from the editor, through a simple point-and-click interface.

This plug-in is bundled by default with *WSMO Studio*.

10.2 Infrawebs Axiom Editor

The INFRAWEBs Axiom editor³ is an ontology-driven tool for graphical construction of WSMML logical expressions. It can be used in two modes: as an extension plug-in of *WSMO Studio* or as a standalone Eclipse application (see [Figure 10.1](#)).

This plug-in is bundled by default with *WSMO Studio*.

The main objective of the editor is to construct valid WSMML logical expressions to be used for describing a capability of WSMO-based Semantic Web Services. The specificity of the underlying representation language is hidden behind rich graphical environment supporting the construction process.

¹This component was developed within the IST project FP6-507483 DIP (<http://dip.semanticweb.org>) by [Barry Norton](#) and [Carlos Pedrinaci](#) from the KMi group of the Open University (<http://kmi.open.ac.uk/>).

²<http://kmi.open.ac.uk/projects/irs/>

³This component was developed within the IST project FP6-511723 Infrawebs (<http://www.infrawebs.org>) by the team of [Prof. Gennady Agre](#) at the Institute of Information Technologies of the Bulgarian Academy of Sciences.

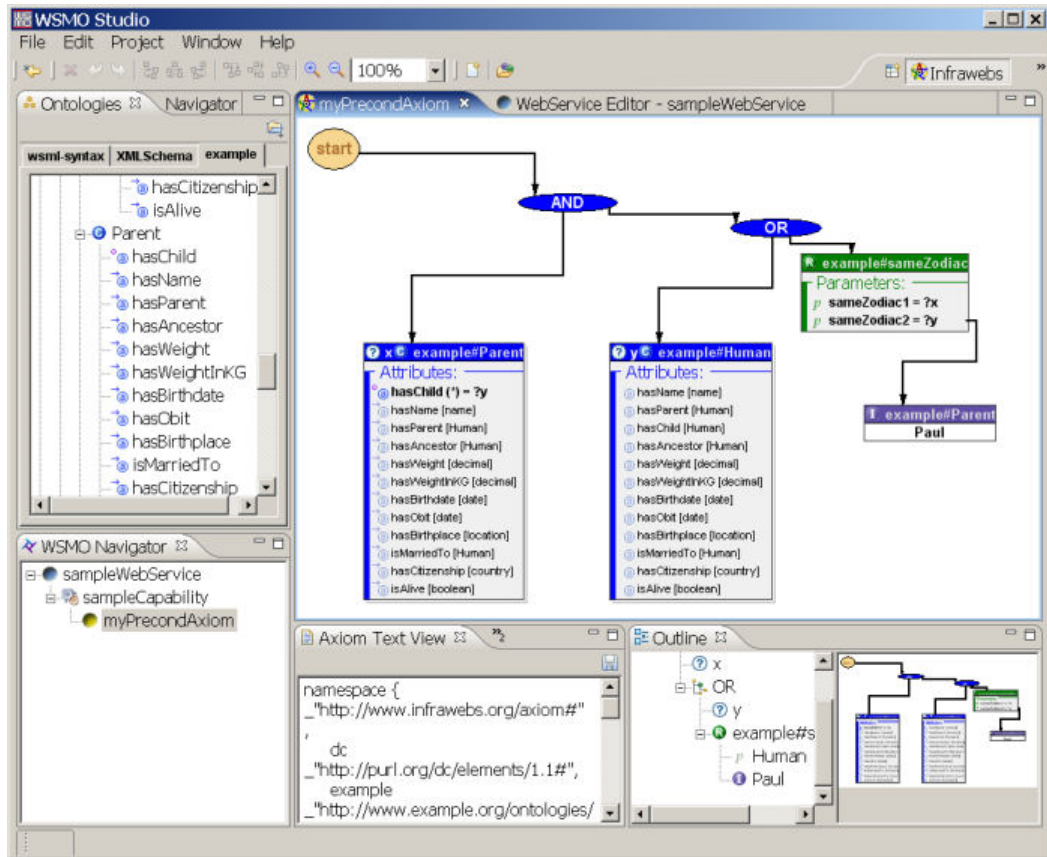


Figure 10.1: Infrawebs Axiom Editor

The editor is comprised of:

- a diagram area (the main editing component)
- and a set of supporting views.

The diagram area contains the graphical representation of logical expression definitions. They are displayed as directed acyclic graphs which reflect their tree structures. By using context menu actions and drag and drop techniques the user can create different diagram nodes and establish connections between them.

The diagram area works in collaboration with an ontology view. The view contains tree representations of loaded WSMO ontologies from which elements can be dragged directly in the diagram area. At each processing step a dedicated text view displays a WSMML fragment generated on the base of the current state of the diagram. Such a preview is generated only for the consistent parts of the diagram. Examples for inconsistencies are not-connected nodes or whole sub-graphs, operator nodes with incomplete set of required operands, etc. When the diagram is complete it can be saved in a WSMML file in the workspace.

10.3.2 Usage

In order to visualise an ontology with the WSMO Visualiser, select from the context menu of the Project Navigator (section 4.5) *Open With* → *WSMO Visualiser* (see Figure 10.3)

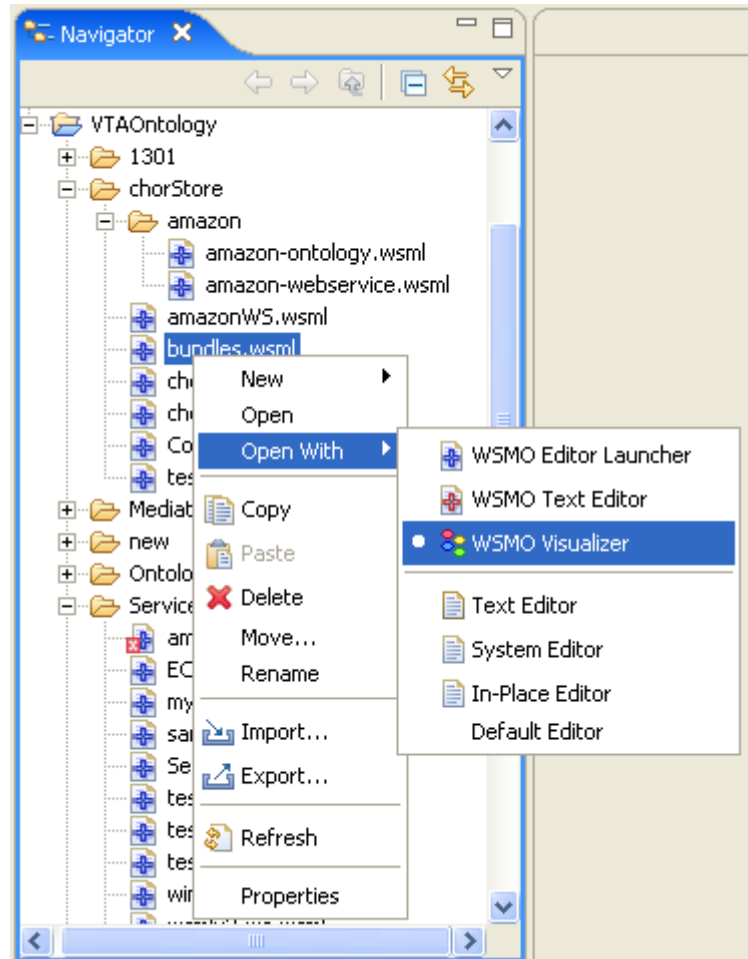


Figure 10.3: Activating the WSMO Visualiser

Chapter 11

Using the Update Site

WSMO Studio supports an *update site* that makes it possible to:

- update existing plug-ins whenever a new version is available
- install newly added plug-ins

In order to update an existing feature or install a new one, one should follow the steps:

- Within *WSMO Studio*, go to *Help* → *Software Updates* → *Find and Install*
- The *WSMO Studio* update site should appear (see [Figure 11.1](#)). If the default update site is not listed, then choose "New Remote Site" and specify <http://www.wsmostudio.org/updates/> as the update site location.
- Proceed, and a list of updates (if any) or new plug-ins will appear ([Figure 11.2](#))
- Choose a feature to install / update and inspect its distribution licence ([Figure 11.3](#))
- Finally, install / update the selected feature ([Figure 11.4](#)) and restart Eclipse.

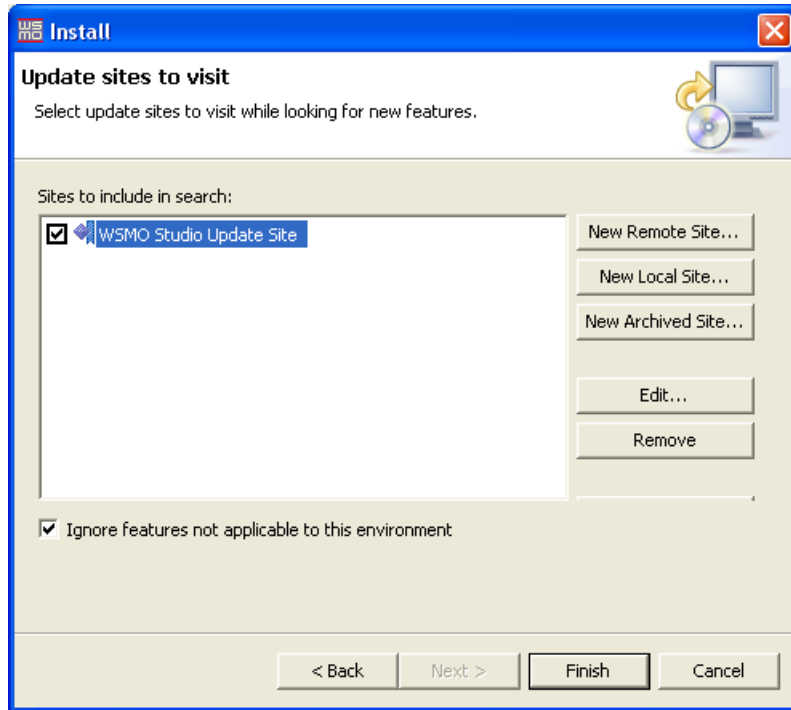


Figure 11.1: Update site – choosing an update site

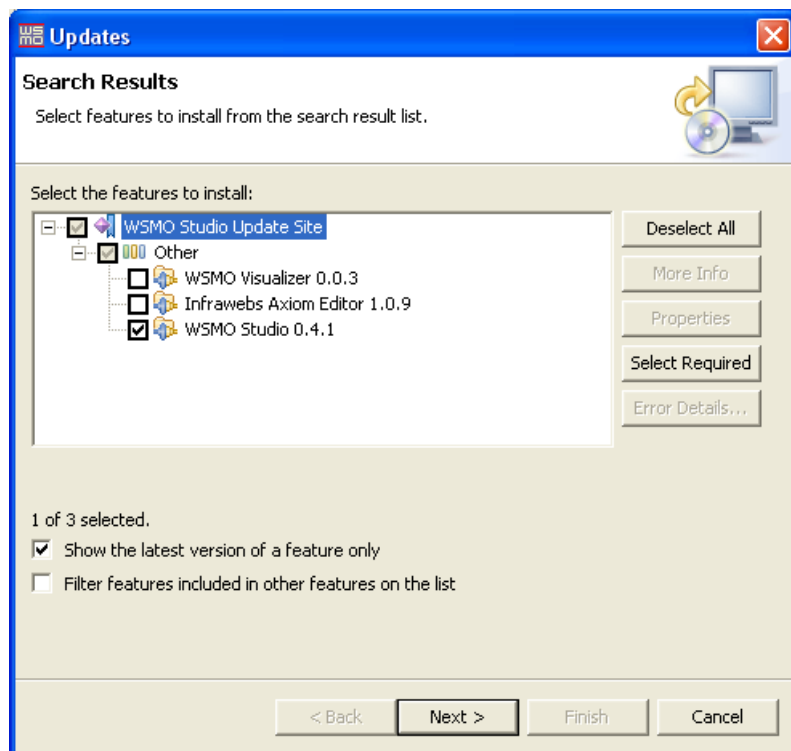


Figure 11.2: Update site – choosing a feature to install

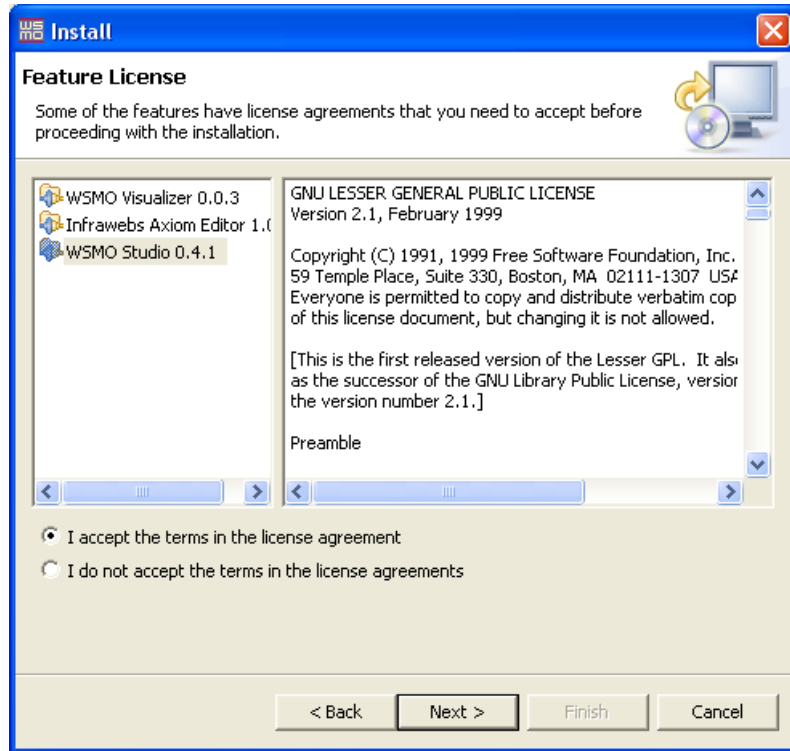


Figure 11.3: Update site – feature licence

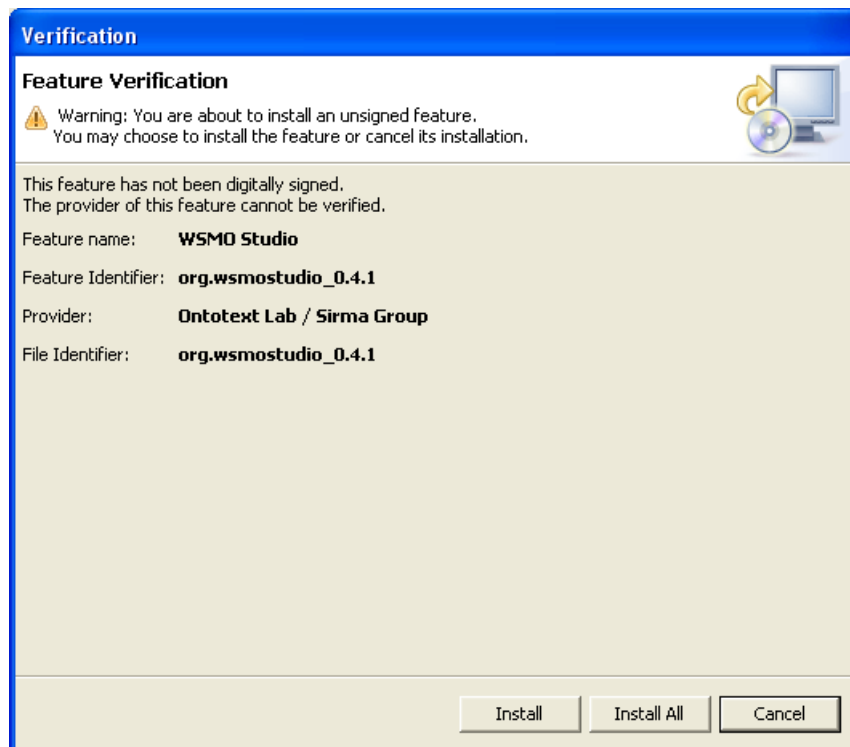


Figure 11.4: Update site – installing a feature

Chapter 12

User Support

The *WSMO Studio* team provides the following user support channels:

- Mailing lists, where questions and problems regarding *WSMO Studio* are discussed – check out <http://www.wsmostudio.org/mail-lists.html> for details
- Bug Tracker, where bugs can be posted – check out http://sourceforge.net/tracker/?group_id=119791&atid=684995
- Request Tracker, where requests for new features and extensions can be submitted: http://sourceforge.net/tracker/?group_id=119791&atid=684998

We also provide a number of RSS feeds that will keep you updated about *WSMO Studio* news and releases at <http://www.wsmostudio.org/rss-feeds.html>

Bibliography

[Cabral & Domingue 05]

L. Cabral and J. Domingue. Mediation of semantic web services in IRS-III. In *International Conference on Service Oriented Computing (ISOC 2005)*, Amsterdam, Netherlands, 2005.

[deBruijn *et al.* 05]

J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. D16.1: WSML family of representation languages. WSML working draft, DERI, October 2005. Available at <http://www.wsmo.org/TR/d16/d16.1/v0.3/>.

[OSGi Alliance 04]

OSGi Alliance. About the OSGi service platform. Technical Whitepaper, July 2004. Available at <http://www.osgi.org>.

[Rivieres & Wiegand 04]

J. D. Rivieres and J. Wiegand. Eclipse: A platform for integrating development tools. *IBM Systems Journal*, 43(2), 2004.

[Scicluna *et al.* 05]

J. Scicluna, A. Polleres, and D. Roman. D14: Ontology-based choreography and orchestration of WSMO. WSMO working draft, DERI, December 2005.

Chapter 13

Appendix A – 3rd Party Components

The following 3rd party components *may* be distributed with *WSMO Studio*¹

component	version	licence	URL
Eclipse	3.1.1	EPL	http://www.eclipse.org
wsmo4j	0.5.2	LGPL	http://wsmo4j.sourceforge.net
ORDI	0.3.0	LGPL	http://www.ontotext.com/ordi/
Sesame	1.2.3	LGPL	http://www.openrdf.org
Wsml2Reasoner	20060307	LGPL	http://dev1.deri.at/wsml2reasoner/
Axis	1.3	ASL	http://ws.apache.org/axis/
commons-dbcp		ASL	http://jakarta.apache.org/commons/dbcp/
commons-el		ASL	http://jakarta.apache.org/commons/el/
commons-logging		ASL	http://jakarta.apache.org/commons/logging/
commons-pool		ASL	http://jakarta.apache.org/commons/pool/

Table 13.1: 3rd party components distributed with *WSMO Studio*

note: *Additional licencing information:*

- *Eclipse Public License*, <http://opensource.org/licenses/eclipse-1.0.php>
- *LGPL*, <http://opensource.org/licenses/lgpl-license.php>
- *Apache Software License*, <http://opensource.org/licenses/apache2.0.php>

¹Depending on your *WSMO Studio* configuration, i.e. the plug-ins that you have installed

Chapter 14

Appendix B – *WSMO Studio* Plug-ins

This section describes the individual *WSMO Studio* plug-ins.

plug-in	wsmo4j
description	Provides a simple wrapper around the wsmo4j library (http://wsmo4j.sourceforge.net) so that it can be exported to other WSMO Studio plug-ins
required / optional	required
Eclipse name	com.ontotext.wsmo4j
SourceForge name	wsmo4j plug-in
version	0.6.0.v20060421

Table 14.1: wsmo4j plug-in

plug-in	WSMO Studio Runtime
description	Provides a core set of functionality that is reused by other plug-ins. It is responsible for the project management in the workspace, the connection with the underlying (WSMO) object model. Additionally, it maintains a set of image icons for the graphical representation of the WSMO objects in the UI
required / optional	required
Eclipse name	org.wsmostudio.runtime
SourceForge name	WSMO Studio Runtime
version	0.4.0

Table 14.2: Runtime plug-in

plug-in	WSMO Editor
description	Provides the WSMO perspective in <i>WSMO Studio</i> , e.g. create WSML descriptions of ontologies, mediators, services, goals, capabilities and interfaces. It comes with a set of UI components for viewing and editing the various WSMO objects
required / optional	required
Eclipse name	org.wsmostudio.ui
SourceForge name	WSMO plug-in
version	0.4.0

Table 14.3: WSMO Editor plug-in

plug-in	Choreography Editor
description	This plug-in provides support for creating WSMO centric choreography descriptions
required / optional	optional
Eclipse name	org.wsmostudio.choreography
SourceForge name	Choreography plug-in
version	0.4.0

Table 14.4: Choreography editor

plug-in	Repository
description	Provides the Repository perspective in <i>WSMO Studio</i> , e.g. working with remote ontology/service/goal repositories. It contains an extension point which allows 3rd parties to supply adapters to their own repositories. At present there are such adapters for a local ORDI (http://www.ontotext.com/ordi/) repository and a Web Service facade to a remote repository
required / optional	optional
Eclipse name	org.wsmostudio.repository
SourceForge name	Repository plug-in
version	0.4.0

Table 14.5: Repository plug-in

plug-in	ORDI adapter
description	This plug-in provides an ORDI adapter to the Repository plug-in and a built-in local repository based on ORDI (http://www.ontotext.com/ordi/)
required / optional	optional
Eclipse name	org.wsmostudio.repository.ordi
SourceForge name	Repository ORDI adapter
version	0.4.0

Table 14.6: ORDI adapter

plug-in	WebService adapter
description	This plug-in provides a WebService facade to a remote Repository. See the relevant WSDL definition http://www.wsmostudio.org/doc/repository.wsdl
required / optional	optional
Eclipse name	org.wsmostudio.repository.webservice
SourceForge name	Repository WebService adapter
version	0.4.0

Table 14.7: Web Service adapter

plug-in	WSML Reasoner adapter
description	This plug-in provides the option of checking ontology consistency by using a Wsml2Reasoner framework (http://dev1.deri.at/wsml2reasoner/)
required / optional	optional
Eclipse name	org.wsmostudio.reasoner.wsml_flight
SourceForge name	WSML reasoner adapter
version	0.4.0

Table 14.8: Web Service adapter

Chapter 15

Appendix C – Licence

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages—typically libraries—of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater

number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not

make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and

distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a

single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Chapter 16

Appendix D – Changelog

Version	Date	Author(s)	Changes
1.8	2006/05/11	marin	new sections added: 1. WSML Validator 2. WSMO Visualiser 3. User Support
1.6	2006/05/10	marin	new section added: Update site
1.5	2006/05/10	marin	new section added: WSML Reasoner
1.4	2006/05/10	alex , marin	new sections added: Infrawebs Axiom Editor and IRS-III adapter
1.3	2006/05/09	alex , marin	1. updated screenshots 2. fixed typos
1.1	2006/05/08	marin	first public draft