DIP
Data, Information and Process Integration with Semantic Web Services
FP6 - 507483

Deliverable

WP 4b: WSMO Platform & Tools
D4.5
WSMO Studio v.1

Marin Dimitrov
Alex Simov
Vassil Momtchev
Martin Hepp
Holger Lausen
Jan Henke
Marc Richardson
Alistair Duke

Jul 15th, 2005
SUMMARY

This deliverable presents the first prototype of an Integrated Service Environment for WSMO, called WSMO Studio. The Studio is 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
- Storing and retrieving WSML descriptions to/from remote service, goal, mediator and ontology repositories

Next versions of this prototype (e.g. DIP D4.11 due M24) will provide improvements of the existing functionality as well as add new functionality related to WSMO centric choreography and orchestration, discovery and interaction with Semantic Web Services runtime environments, such as WSMX.

This deliverable directly contributes to one of the major DIP goals: exploitable tools. In our previous work (DIP D4.4) we have stressed the important role that robust and easy-to-use end user tools play in the adoption of any technological innovation (such as WSMO), and the goal of this deliverable is to provide a GUI tool that assists users working in the WSMO domain with the tasks related to semantic web service annotation – creating WSML descriptions of ontologies, goals, services and mediators, storing and retrieving WSMO elements to/from ontology, service, mediator and goal repositories.

WSMO Studio is directly relevant to the case study work packages of the DIP project where semantic annotation (in a WSMO centric way) of web services is required. WSMO Studio is based on a modular and extensible architecture which makes it possible to integrate prototypes (for example ontology versioning tools, ontology management tools, ontology mediation components, etc.) developed within other DIP work packages – WP1, WP4, WP5 and WP6.

The target audience of this deliverable consists of:

- **End users** that will use WSMO Studio to work with WSMO service descriptions. The sections that explain the user interface and the basic functionality of WSMO Studio are intended for such readers.
- **Solution providers** that provide new functionality relevant to WSMO/WSMX and that would like to integrate this functionality with WSMO Studio. These readers should focus on the sections that contain technical details of the WSMO Studio architecture and specifically the extension point definitions that make it possible for new functionality to be dynamically integrated within the tool.

This work is partially related to another effort for producing a set of GUI tools for WSMO – the Web Services Modelling Toolkit part of the WSMX toolset. Although WSMO Studio and WSMT share some common goals (e.g. providing an easy-to-use GUI for users working with WSMO), the architectural and design approaches at present differ substantially.
Disclaimer: The DIP Consortium is proprietary. There is no warranty for the accuracy or completeness of the information, text, graphics, links or other items contained within this material. This document represents the common view of the consortium and does not necessarily reflect the view of the individual partners.
**Abstract (for dissemination)**

This deliverable presents the first prototype of such an ISE, based on Eclipse, called WSMO Studio that provides the following functionality:

- Creating WSMO descriptions of ontologies, goals, web services and mediators
- Export and import of the WSMO descriptions
- Storing and retrieving WSML descriptions to/from remote service, goal, mediator and ontology repositories

**Keywords**

Integrated Service Environment, WSMO, WSML, Eclipse, Java
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-05-2005</td>
<td>001</td>
<td>Marin Dimitrov</td>
<td>Initial structure of the document (TOC)</td>
</tr>
<tr>
<td>20-06-2005</td>
<td>002</td>
<td>Marin Dimitrov</td>
<td>Version released for internal review</td>
</tr>
<tr>
<td>11-07-2005</td>
<td>003</td>
<td>Marin Dimitrov</td>
<td>Corrections based on 1st internal review</td>
</tr>
<tr>
<td>15-07-2005</td>
<td>004</td>
<td>Marin Dimitrov</td>
<td>Corrections based on 2nd internal review</td>
</tr>
</tbody>
</table>
## Project Consortium Information

<table>
<thead>
<tr>
<th>Partner</th>
<th>Acronym</th>
<th>Contact</th>
</tr>
</thead>
</table>
| National University of Ireland Galway | NUIG | Prof. Dr. Christoph Bussler  
Digital Enterprise Research Institute (DERI)  
National University of Ireland, Galway  
Galway  
Ireland  
Email: chris.bussler@deri.org  
Tel: +353 91 512460 |
| Fundacion De La Innovacion.Bankinter | Bankinter | Monica Martinez Montes  
Fundacion de la Innovation. Bankinter  
Paseo Castellana, 29  
28046 Madrid,  
Spain  
Email: mmtnez@bankinter.es  
Tel: 916234238 |
| Berlecon Research GmbH | Berlecon | Dr. Thorsten Wichmann  
Berlecon Research GmbH  
Oranienburger Str. 32  
10117 Berlin,  
Germany  
Email: tw@berlecon.de  
Tel: +49 30 2852960 |
| British Telecommunications Plc. | BT | Dr John Davies  
BT Exact (Orion Floor 5 pp12)  
Adastral Park Martlesham  
Ipswich IP5 3RE,  
United Kingdom  
Email: john.nz.davies@bt.com  
Tel: +44 1473 609583 |
| Swiss Federal Institute of Technology, Lausanne | EPFL | Prof. Karl Aberer  
Distributed Information Systems Laboratory  
École Polytechnique Fédérale de Lausanne  
Bât. PSE-A  
1015 Lausanne, Switzerland  
Email: Karl.Aberer@epfl.ch  
Tel: +41 21 693 4679 |
| Essex County Council | Essex | Mary Rowlett,  
Essex County Council  
PO Box 11, County Hall, Duke Street  
Chelmsford, Essex, CM1 1LX  
United Kingdom.  
Email: maryr@essexcc.gov.uk  
Tel: +44 (0)1245 436524 |
| Forschungszentrum Informatik | FZI | Andreas Abecker  
Forschungszentrum Informatik  
Haid-und-Neu Strasse 10-14  
76131 Karlsruhe  
Germany  
Email: abecker@fzi.de  
Tel: +49 721 9654 0 |
<table>
<thead>
<tr>
<th>Partner</th>
<th>Acronym</th>
<th>Contact</th>
</tr>
</thead>
</table>
| Institut für Informatik, Leopold-Franzens Universität Innsbruck | UIBK | Prof. Dieter Fensel  
Institute of computer science  
University of Innsbruck  
Technikerstr. 25  
A-6020 Innsbruck, Austria  
Email: dieter.fensel@deri.org  
Tel: +43 512 5076485 |
| ILOG SA | ILOG | Christian de Sainte Marie  
9 Rue de Verdun, 94253  
Gentilly, France  
Email: csma@ilog.fr  
Tel: +33 1 49082981 |
| inubit AG | inubit | Torsten Schmale  
inubit AG  
Lützowstraße 105-106  
D-10785 Berlin  
Germany  
Email: ts@inubit.com  
Tel: +49 30726112 0 |
| Intelligent Software Components, S.A. | iSOCO | Dr. V. Richard Benjamins, Director R&D  
Intelligent Software Components, S.A.  
Pedro de Valdivia 10  
28006 Madrid, Spain  
Email: rbenjamins@isoco.com  
Tel. +34 913 349 797 |
| NIWA WEB Solutions | NIWA | Alexander Wahler  
NIWA WEB Solutions  
Niederacher & Wahler OEG  
Kirchengasse 13/1a  
A-1070 Wien  
Email: wahler@niwa.at  
Tel:+43(0)1 3195843-11 |
| The Open University | OU | Dr. John Domingue  
Knowledge Media Institute  
The Open University, Walton Hall  
Milton Keynes, MK7 6AA  
United Kingdom  
Email: lb.domingue@open.ac.uk  
Tel.: +44 1908 655014 |
| SAP AG | SAP | Dr. Elmar Dorner  
SAP Research, CEC Karlsruhe  
SAP AG  
Vincenz-Priessnitz-Str. 1  
76131 Karlsruhe, Germany  
Email: elmar.dorner@sap.com  
Tel: +49 721 6902 31 |
<table>
<thead>
<tr>
<th>Company</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sirma AI Ltd.</td>
<td>Atanas Kiryakov, Ontotext Lab, - Sirma AI EAD&lt;br&gt;Office Express IT Centre, 3rd Floor&lt;br&gt;135 Tzarigradsko Chausse&lt;br&gt;Sofia 1784, Bulgaria&lt;br&gt;Email: <a href="mailto:atanas.kiryakov@sirma.bg">atanas.kiryakov@sirma.bg</a>&lt;br&gt;Tel.: +359 2 9768 303</td>
</tr>
<tr>
<td>Unicorn Solution Ltd.</td>
<td>Jeff Eisenberg&lt;br&gt;Unicorn Solutions Ltd,&lt;br&gt;Malcha Technology Park 1&lt;br&gt;Jerusalem 96951&lt;br&gt;Israel&lt;br&gt;Email: <a href="mailto:Jeff.Eisenberg@unicorn.com">Jeff.Eisenberg@unicorn.com</a>&lt;br&gt;Tel.: +972 2 6491111</td>
</tr>
<tr>
<td>Vrije Universiteit Brussel</td>
<td>Pieter De Leenheer&lt;br&gt;Starlab- VUB&lt;br&gt;Vrije Universiteit Brussel&lt;br&gt;Pleinlaan 2, G-10&lt;br&gt;1050 Brussel ,Belgium&lt;br&gt;Email: <a href="mailto:Pieter.De.Leeheer@vub.ac.be">Pieter.De.Leeheer@vub.ac.be</a>&lt;br&gt;Tel.: +32 (0) 2 629 3749</td>
</tr>
</tbody>
</table>
# LIST OF KEY WORDS/ABBREVIATIONS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>ISE</td>
<td>Integrated Service Environment</td>
</tr>
<tr>
<td>J2SE</td>
<td>Java 2 Platform Standard Edition</td>
</tr>
<tr>
<td>NFP</td>
<td>Non-Functional Property</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>LGPL</td>
<td>GNU Lesser General Public Licence</td>
</tr>
<tr>
<td>SWS</td>
<td>Semantic Web Services</td>
</tr>
<tr>
<td>WSML</td>
<td>Web Service Modeling Language</td>
</tr>
<tr>
<td>WSMO</td>
<td>Web Service Modeling Ontology</td>
</tr>
<tr>
<td>WSMT</td>
<td>Web Service Modeling Toolkit</td>
</tr>
<tr>
<td>WSMX</td>
<td>Web Service Execution Environment</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS

SUMMARY .......................................................................................................................... I
LIST OF KEY WORDS/ABBREVIATIONS ................................................................. VIII
TABLE OF CONTENTS ................................................................................................. IX

1 INTRODUCTION .............................................................................................................. 1
2 USERS GUIDE ................................................................................................................. 2
   2.1 Introduction to Eclipse ........................................................................................... 2
   2.2 Installing WSMO Studio ........................................................................................ 3
   2.3 The WSMO Studio UI ............................................................................................ 3
      2.3.1 Standalone vs. Embedded Usage .................................................................. 3
      2.3.2 Menus ............................................................................................................ 4
      2.3.3 Wizards .......................................................................................................... 5
      2.3.4 Perspectives .................................................................................................. 6
   2.4 The WSMO Perspective .......................................................................................... 6
      2.4.1 Project Navigator .......................................................................................... 7
      2.4.2 WSMO Navigator .......................................................................................... 8
      2.4.3 Editors ........................................................................................................... 9
   2.5 The Repository Perspective .................................................................................. 14
      2.5.1 Repository Explorer ...................................................................................... 14
      2.5.2 Repository Views .......................................................................................... 15

3 DEVELOPERS GUIDE ................................................................................................... 18
   3.1 Architecture ............................................................................................................ 18
   3.2 WSMO Studio Plug-ins ......................................................................................... 19
   3.3 Extension Points ..................................................................................................... 20
   3.4 Repository Interface ............................................................................................... 21

4 CONCLUSION ............................................................................................................... 22

REFERENCES .................................................................................................................. 22

5 APPENDIX A – INSTALLATION INSTRUCTIONS ......................................................... 23
   5.1 Requirements ......................................................................................................... 23
      5.1.1 Operating Systems ......................................................................................... 23
      5.1.2 Java ................................................................................................................ 24
      5.1.3 Eclipse ............................................................................................................ 24
   5.2 Installation ............................................................................................................. 24
5.2.1 Standalone mode ........................................................................................................ 24
5.2.2 Embedded mode ........................................................................................................ 24

6 APPENDIX B – TECHNICAL DOCUMENTATION .......................................................... 25
6.1 JavaDoc ......................................................................................................................... 25
6.2 Source files .................................................................................................................... 25
6.3 Plug-in Descriptors ........................................................................................................ 26
  6.3.1 Wsmo4j Plug-in ....................................................................................................... 26
  6.3.2 WSMO Studio Runtime ......................................................................................... 26
  6.3.3 WSMO Plug-in ........................................................................................................ 27
  6.3.4 Repository plug-in .................................................................................................. 32
6.4 Eclipse Extension Points ............................................................................................. 33
  6.4.1 Repository extensions ............................................................................................ 33
  6.4.2 WSMO Editors ....................................................................................................... 34
6.5 Repository Interface ..................................................................................................... 35

7 APPENDIX C – LICENCE ................................................................................................ 35
7.1 WSMO Studio Licence ................................................................................................ 36
7.2 3rd Party Dependencies and Licences ........................................................................ 36

LIST OF FIGURES
Figure 1 Wizard selection dialog ......................................................................................... 5
Figure 2 Default perspectives ............................................................................................. 6
Figure 3 Project Navigator .................................................................................................. 7
Figure 4 WSMO Navigator – Ontology ............................................................................... 8
Figure 5 WSMO Navigator - Web Service ......................................................................... 9
Figure 6 Non-functional properties editor ......................................................................... 10
Figure 7 Attribute Editor ................................................................................................... 11
Figure 8 Instance editor ..................................................................................................... 12
Figure 9 Ontology editor .................................................................................................... 13
Figure 10 Importing web services into a repository .......................................................... 16
Figure 11 Confirmation dialog ......................................................................................... 16
Figure 12 Repository perspective ..................................................................................... 17
Figure 13 WSMO Studio within the Eclipse platform (from [1]) ....................................... 19
Figure 14 WSMO Repository façade ............................................................................... 21
Figure 15 Repository interface ......................................................................................... 22
LIST OF TABLES
Table 1 Eclipse concepts .................................................................................................. 3
Table 2 Plug-in descriptions........................................................................................... 20
Table 3 3rd party software part of WSMO Studio .......................................................... 36
1 INTRODUCTION

The role of WP4b in DIP is to provide a set of tools for using and exploiting semantically described Web Services. Our previous work on [1] presented a detailed requirements analysis and a specification for an Integrated Service Environment (ISE) for WSMO\(^1\). We outlined why certain non-functional requirements such as *role-oriented development*, *extensibility*, adherence to *open standards*, *usability* and *flexible licensing* are important and how they influence the architecture of an ISE. We have also stressed the important role that robust and easy-to-use end user tools play in the adoption of any technological innovation (such as WSMO). The goal of this deliverable is to provide a GUI tool that assists the users working in the WSMO domain with tasks related to semantic web service annotation.

This deliverable presents the first prototype of such an ISE, called *WSMO Studio*\(^2\). The Studio is a standalone, Eclipse\(^3\) based application that supplies the following functionality:

- Creating WSMO descriptions of ontologies, goals, web services and mediators
- Export and import of the WSMO descriptions
- Storing and retrieving WSML descriptions to/from remote service, goal, mediator and ontology repositories

Future versions of this prototype will provide improvements of the existing functionality as well as new functionality related to WSMO centric choreography and orchestration, discovery and interaction with Semantic Web Services runtime environments such as WSMX.

The target audience of this deliverable consists of:

- *End users* that will use WSMO Studio to work with WSMO service descriptions. The sections that explain the user interface and the basic functionality of WSMO Studio are intended for such readers.

- *Solution providers* that provide new functionality relevant to WSMO/WSMX and that would like to integrate this functionality with WSMO Studio. These readers should focus on the sections that contain technical details of the WSMO Studio architecture and specifically the extension point definitions that make it possible for new functionality to be dynamically integrated within the tool.

This document is structured as follows:

---

1 http://www.wsmo.org
2 http://www.wsmostudio.org
3 http://www.eclipse.org
• Section 2 presents a short introduction to the GUI and the way to use its functionality. This section is intended for the end-users of WSMO Studio that will create semantic annotations for web services in a WSMO centric way.

• Section 3 is intended for developers that will extend the existing functionality of WSMO Studio or will provide new functionality (by means of new plug-ins that can be integrated with the Studio). Technical details on the architecture of the Studio and a summary of the extension points are available in this section.

• Section 5 contains the prerequisites for installing WSMO Studio as well as the installation steps.

• Section 6 is again intended for developers and contains more technical details relevant to section 3.

• Finally, section 7 contains the licensing terms of WSMO Studio and the 3rd party software components part of the Studio distribution.

2 USERS GUIDE

This section contains a brief introduction to the basic Eclipse concepts as well as a user guide for WSMO Studio (including detailed description of the provided functionality and screenshots).

2.1 Introduction to Eclipse

The specification of WSMO Studio [1] already contains an introduction to the basic Eclipse concepts from a developer’s perspective. We will briefly reiterate these concepts from the user’s point of view, so that the end-user, who is unfamiliar with Eclipse, may be acquainted with Eclipse related concepts and terms that are used throughout this document.

Table 1 contains a summary of the basic Eclipse concepts that the end-user should be acquainted with.
### Table 1 Eclipse concepts

<table>
<thead>
<tr>
<th>Concept/Term</th>
<th>Description</th>
<th>Relation to \textit{WSMO Studio}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plug-in</td>
<td>In Eclipse, all the functionality is encapsulated in the form of plug-ins</td>
<td>The functionality of \textit{WSMO Studio} is available as a set of Eclipse plug-ins(^5)</td>
</tr>
<tr>
<td>Workspace, projects, files</td>
<td>Workspaces in Eclipse provide context for resources (files, folders, etc.). Projects group together related resources</td>
<td>Projects in \textit{WSMO Studio} group together a set of related WSML descriptions. The latter may be represented either as files on the local filesystem or reside in a remote WSMO repository</td>
</tr>
<tr>
<td>Views, editors and perspectives</td>
<td>Views and editors in Eclipse are containers for logically and functionally related UI sub-components (text areas, buttons, etc.). Perspectives are containers for logically and functionally related views and editors. Only one perspective is visible at any moment but several perspectives may be active.</td>
<td>Every WSMO element (ontology, web service, goal, mediator) is associated with specific views and editors in \textit{WSMO Studio}. There are two perspectives defined in \textit{WSMO Studio} at present: \textit{WSMO perspective} and \textit{Repository perspective}.</td>
</tr>
</tbody>
</table>

2.2 Installing WSMO Studio

The prerequisites for installing \textit{WSMO Studio} as well as the detailed installation steps are available in Appendix A – Installation Instructions.

2.3 The \textit{WSMO Studio} UI

This section contains an overview of the basic UI elements of \textit{WSMO Studio}.

2.3.1 Standalone vs. Embedded Usage

\(^4\) … with the exception of the micro-kernel that manages the plug-ins

\(^5\) See section 3.2 for more details on the plug-ins available in \textit{WSMO Studio}
WSMO Studio can be used in two modes:

- **As a standalone application** – this is the case when the user downloads\(^6\) the Studio as a pre-packaged application that includes the Eclipse runtime and core plug-ins as well as the *WSMO Studio* plug-ins. We recommend this mode for users who are either novice to Eclipse or use Eclipse only for running *WSMO Studio*.

- **As part of an existing Eclipse installation** – this is the case when the user downloads only the relevant WSMO Studio plug-ins and copies them into the ECLIPSE/plugins folder of Eclipse. This mode is recommended for expert users who already use Eclipse and would like to only add the *WSMO Studio* functionality to an existing Eclipse deployment.

The following sections explain the WSMO Studio UI from the point of view of a standalone WSMO Studio application. Concrete menus, wizards and layouts may differ for the embedded mode, depending on the installed plug-ins and the user specified Eclipse preferences, but the overall UI will not differ substantially between the two modes.

### 2.3.2 Menus

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

1. **File**
   a. **New** – starts a Wizard for creating (see Figure 1):
      i. WSMO Projects
      ii. Ontologies
      iii. Goals
      iv. Capabilities
      v. Interfaces
      vi. Web Services
      vii. Mediators
   b. **Save / Save All** – saves the modified WSML descriptions
   c. **Exit** – exits the application. Depending on the user’s preferences, a confirmation dialog may appear.

2. **Edit** – standard editing functionality (copy & paste, undo & redo, etc.)
3. **Project** – open / close projects
4. **Window** – standard Eclipse settings for perspectives, views and preferences

\(^6\) *WSMO Studio* can be downloaded at [http://www.wsmostudio.org/download.html](http://www.wsmostudio.org/download.html)
5. Help – help subsystem and about box. WSMO Studio at present does not provide integrated help contents.

![Figure 1 Wizard selection dialog](image)

2.3.3 Wizards

The following wizards are available by default in *WSMO Studio*\(^7\) (see Figure 1):

- WSMO Project wizard
- Ontology wizard
- Goal wizard
- Capability wizard
- Interface wizard
- Web Service 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. At present wizards do not provide any added value, since they do not assist the user in a step-by-step creation of WSML.

---

\(^7\) Future versions may provide additional wizards.
descriptions, but future versions may provide such extended wizards. More functional wizards can also be plugged-in by third parties, using standard Eclipse extension mechanisms.

2.3.4 Perspectives

*WSMO Studio* defines two perspectives at present⁸:

- *WSMO perspective*, and
- *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 formats⁹. More information about the *WSMO perspective* is available in section 2.4.

The *Repository perspective* groups together functionality for accessing remote ontology, goal, service and mediator repositories. More information is available in section 2.5.

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

![Figure 2 Default perspectives](image)

### 2.4 The *WSMO* Perspective

When creating descriptions of WSMO elements, the user works in the *WSMO perspective* (the perspective is activated by default when a user opens an existing WSML file or creates a new element with the help of a wizard).

---

⁸ Although the “WSMO Studio Requirements & Specification” document outlined the three perspectives - Ontology, WSMO and Repository, during the implementation phase, we have decided to merge together the first two perspectives since they were too closely related.

⁹ Only the default WSML format is supported at present but work is in progress for supporting the XML/RDF representation of WSML as well (refer to [2] for more details on the various WSML formats)
The perspective consists of the following views and editors\textsuperscript{10}:

- Project navigator, usually located in the top-left part of the UI area\textsuperscript{11}
- WSMO navigator, located in the bottom-left part
- 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.

### 2.4.1 Project Navigator

The Project Navigator (see Figure 3) presents a tree view of the projects in the current workspace, together with the WSML files associated with a project. Double clicking a WSML file from a project will activate the respective views and editors associated with the type of WSMO entity (Ontology, Web Service, Mediator, Goal) described in the file. Documents can also be activated from the context menu (see Figure 3).

![Figure 3 Project Navigator](image)

New descriptions may be created as well with the help of the New action of the context menu, which will activate the respective wizard\textsuperscript{12} (see Figure 1).

\textsuperscript{10} Note that a user may hide/show a view or an editor from a perspective at any time.

\textsuperscript{11} Note that a user may change the exact location of a visual component as well, so that the exact location of any WSMO Studio UI element may differ, depending on the user’s preferences.

\textsuperscript{12} The New action of the context menu should list the available WSMO wizards. If there are no wizards listed by default, then one should choose New --> Other, and the wizards will be listed under the WSMO
2.4.2 WSMO Navigator

The *WSMO Navigator* (see Figure 4 and Figure 5) presents an entity specific summary view of the selected WSMO entity.

If the entity is an ontology then the concept / relation hierarchy is shown (together with the instances of each concept / relation) 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 4 WSMO Navigator – Ontology](image)

section of the wizard selection dialog. To ensure that the WSMO wizards are listed by default one should choose from the main menu *Window -> Customise Perspective -> Shortcuts* and make sure that for the “New” submenu the WSMO shortcut category is checked.
If the WSMO entity is a Web Service then WSMO Navigator shows the capability and the list of interfaces associated with the Web Service (see Figure 5). The context menu is again element specific.

![Figure 5 WSMO Navigator - Web Service](image)

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

### 2.4.3 Editors

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

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

- Identifier\(^{13}\)
- Non-functional properties (see Figure 6)
- Used namespaces
- Imported ontologies
- Used mediators

---

\(^{13}\) Note that identifiers are immutable, e.g. once specified they cannot be modified.
Figure 6 Non-functional properties editor

Figure 9 shows the UI components for a WSMO ontology.
The following sections will present more details on specific editors and views.

2.4.3.1 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 9).

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

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

These specific editors are activated when an element from the WSMO Navigator (Figure 4) is chosen for editing, either by double clicking on the respective element or by choosing “Edit...” from the context menu.

We’ll briefly summarise the functionality of the ontology related editors, without too many additional screenshots.

**Concept / Relation Editor**
The concept editor provides means for specifying an ontology concept and its properties as defined in [4] and [2]:

- Attributes: add / remove attributes (see Figure 7)
- Super concepts: add / remove super concepts
- Instances: add / remove instances

![Attribute Editor](image)

**Figure 7 Attribute Editor**

**Instance Editor**

The instance editor (Figure 8) provides the following functionality:

- Specify the concepts that the instance belongs to
- Specify attribute values for the attributes defined for the concept
The Axiom Editor provides means for specifying the logical expressions that define an axiom. At present, the 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. See section 3.3 for details.
2.4.3.2 Goal / Web Service Editors

Figure 9 Ontology editor
In addition to the common UI elements (see section 2.4.3), the Goal / Web Service editors provide means for specifying the capability and the list of interfaces associated with a Goal / Web Service.

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

The Interface Editor is not fully functional at present since WSMO choreography and orchestration specifications ([5] and [6]) are not fully specified at present. A fully functional Orchestration and Choreography Editor will be provided with the second version of WSMO Studio\(^\text{15}\).

### 2.4.3.3 Mediator Editors

The Mediator Editors provide all the common UI functionality plus means to specify the source and target components of a mediator and the mediation service URI.

### 2.5 The Repository Perspective

The **Repository perspective** (Figure 12) groups together functionality for accessing remote ontology, goal, service and mediator repositories\(^\text{16}\).

The UI area of the Repository perspective contains several components:

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

#### 2.5.1 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

\(^{15}\) 3rd parties may provide Interface editors even before the 2nd release of WSMO Studio since the Interface Editor is an extension point. See section 3.3 for details.

\(^{16}\) Note that in order to be accessible from WSMO Studio, the remote repository should provide a façade interface that WMSO Studio can interact with. More details are available in section 3.4
remote repository, the list of ontologies, mediators, services and goals will be populated in the respective UI components (see next section).

2.5.2 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, see section 2.4.1) 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

The sequence of steps that the user should follow in order to be able to store WSML descriptions in a remote repository is:

1. Create a WSML description using the WSMO perspective (see previous chapter) and save it in a project in the workspace
2. Switch to the repository perspective
3. Connect to a remote repository
4. Import the WSML file into the remote repository (Import from Workspace from the context menu, see Figure 10)
Figure 10 Importing web services into a repository

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 to the WSMO perspective
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 WSML description from the local workspace back to the remote repository (Import from Workspace from the context menu, see Figure 10). At this point, the user will be prompted for confirmation whether to overwrite the WSML description that already exist in the repository (see Figure 11)

Figure 11 Confirmation dialog
Figure 12 Repository perspective
3 DEVELOPERS GUIDE

The section is intended for developers that wish to contribute to WSMO Studio by either:

- extend the provided functionality (by means of providing extensions to the published extension points), or
- provide new functionality (by providing new plug-ins)

The section contains a brief overview of the WSMO Studio architecture, descriptions of the provided plug-ins, extension points and the repository interface used by the Repository perspective. Concrete technical details (such as plug-in descriptors, Java interfaces, WSDL interfaces) are omitted from this section and are available in Appendix B – Technical Documentation.

3.1 Architecture

Within the Eclipse platform, all the functionality (with the exception of the micro-kernel17) is provided by means of plug-ins. WSMO Studio follows the Eclipse approach as the application consists of a set of plug-ins (see section 3.2).

17 i.e. the part of Eclipse that manages other plug-ins.
3.2 WSMO Studio Plug-ins

Table 2 presents the plug-ins that make up *WSMO Studio* at present. Individual plug-in descriptors are available in section 6.3. The following section contains details on the plug-in extension points (if defined).
### Table 2 Plug-in descriptions

<table>
<thead>
<tr>
<th>Plug-in</th>
<th>ID</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
</table>
| wsmo4j plug-in             | com.ontotext.wsmo4j          | Provides a simple wrapper of the wsmo4j library so that it is available to other plug-ins | • Mandatory  
• No extensions  
• No dependencies |
| WSMO Studio Runtime        | org.wsmostudio.runtime       | Provides common functionality that is used by other WSMO Studio plugins     | • Mandatory  
• Extensible  
• Depends on:  
  o core Eclipse plug-ins  
  o Wsmo4j plug-in          |
| WSMO plug-in               | org.wsmostudio.ui            | Functionality for the WSMO perspective                                      | • Mandatory  
• Extensible  
• Depends on:  
  o core Eclipse plug-ins  
  o WSMO Studio runtime  
  o Wsmo4j plug-in          |
| Repository plug-in         | org.wsmostudio.repository    | Functionality for the Repository perspective                               | • Optional  
• Extensible  
• Depends on:  
  o core Eclipse plug-ins  
  o WSMO Studio runtime  
  o Wsmo4j plug-in          |
|                            | org.wsmostudio.repository.webservice |                                                                     |                            |
| Application                | org.wsmostudio.application   | Customisation of the standalone application (not required for the embedded usage mode) | • Optional  
• No extensions  
• Depends on:  
  o core Eclipse plug-ins |

### 3.3 Extension Points

The following extension points are defined at present:

---

18 [http://wsmo4j.sourceforge.net](http://wsmo4j.sourceforge.net)
1. **Axiom editor** – part of the WSMO plug-in, the purpose of this extension point is to allow custom logical expression editors to be plugged-in, since the Logical Expression Editor at present provides very limited functionality. See the WSMO plug-in descriptor (section 6.3.1) for more details.

2. **Interface editor** – since WSMO centric orchestration and choreography specifications are not finalised at this point, it is not possible at present to create such descriptions (part of the WSMO service interface). The second version of the prototype will provide such an editor, but meanwhile 3rd parties may use this extension point to plug-in their custom implementations related to orchestration and choreography descriptions.

3. **Repository** – this extension point is intended to provide a pluggable bridge to various ontology, service, goal and mediator repository facades. **WSMO Studio** comes by default with such a bridge that uses SOAP messages to communicate with the remote repository façade (see section 6.5 for the WSDL definitions). See the Repository plug-in descriptor (6.3.4) for more details and section 3.4 for details on the Repository interface.

### 3.4 Repository Interface

To allow **WSMO Studio** to interact with various ontology/goal/service repositories we have provided an additional abstraction layer that isolates **WSMO Studio** from the particular details (in terms of API) of the various repositories. We use a façade called **Repository** that hides the specific ontology, mediator, service and goal repositories and provides a unified interface to communicate with. Repository providers that wish to be able to interact with **WSMO Studio** should provide an implementation of this interface.

Figure 14 presents the interaction between **WSMO Studio** and the WSMO Repository façade. Note that the repository façade may aggregate data (e.g. WSML descriptions) from more than one ontology/mediator/goal/service repository.

---

19 Part of wsmo4j (http://wsmo4j.sourceforge.net)

20 **WSMO Studio** comes by default with such an implementation that allows WSML descriptions to be stored on a centralised filesystem based repository.
Figure 15 presents the Repository interface.

```
interface DataStore

+load(id:Identifier):Entity
+save(item:Entity):void
+remove(id:Identifier):void

interface WsmoRepository

+getDescription():String
+setDescription(desc:String):void
+getVersion():String
+saveOntology(ont:Ontology):void
+deleteOntology(ont:Ontology):void
+getOntology(id:IRI):Ontology
+deleteOntology(id:IRI):void
+listOntologies():List
+addOntology(ont:Ontology):void
+saveOntology(ont:Ontology):void
+getOntology(id:IRI):Ontology
+deleteOntology(id:IRI):void
+listOntologies():List
+addGoal(goal:Goal):void
+saveGoal(goal:Goal):void
+deleteGoal(id:IRI):void
+listGoals():List
+addMediator(med:Mediator):void
+saveMediator(med:Mediator):void
+deleteMediator(id:IRI):void
+listMediators():List
+addWebService(ws:WebService):void
+saveWebService(ws:WebService):void
+deleteWebService(id:IRI):void
+listWebServices():List
```

Figure 15 Repository interface

4 CONCLUSION

This deliverable presented the first prototype of an Integrated Service Environment for WSMO, called WSMO Studio. The deliverable contains information for both end users (UI guideline, installation instructions) and 3rd party developers intending to provide additional functionality (extension points, interface definitions, plug-in descriptors).

Future work on D4.11 (due M24) will provide an improved version of the WSMO Studio prototype that includes new functionality related to service orchestration / choreography and interaction with SWS runtime environments (such as WSMX).

REFERENCES
5 APPENDIX A – INSTALLATION INSTRUCTIONS

5.1 Requirements

This section contains the detailed requirements for installing WSMO Studio.

5.1.1 Operating Systems

The Eclipse project page contains a list of the supported operating systems\(^\text{21}\). In summary, the following OS (incl. processor type and window manager) are supported:

- Windows
- Linux (x86 / Motif or GTK2)
- Linux (AMD-64 or IA-64 / GTK2)
- OSX (Mac / Carbon)
- Solaris 8 (Sparc / Motif or GTK2)
- AIX (PPC / Motif)
- HP-UX (HP9000 or IA-64 / Motif)

5.1.2 Java

WSMO Studio is a J2SE application, and it requires that a JRE is already present on the machine. If JRE is not already installed then it can be obtained from http://java.sun.com (or an alternative location, depending on the operating system).

WSMO Studio requires JRE version 1.4.2 or later.\(^{22}\)

5.1.3 Eclipse

WSMO Studio is compliant with Eclipse 3.1 or later. Note that if you are using the standalone version of WSMO Studio then Eclipse is bundled with it.

5.2 Installation

WSMO Studio can be used in two modes: as a standalone application or as part of an existing Eclipse deployment. Before installing, make sure that you are acquainted with the licensing terms (see Appendix C – Licence)

5.2.1 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.\(^{23}\)

Distributions of WSMO Studio for Windows, Linux (x86 / GTK2) and OSX are available at the web site. If you use a different OS or a configuration for which there is no standalone distribution then you should download Eclipse for the respective configuration and install the individual plug-ins as explained in section 5.2.2

5.2.2 Embedded mode

If you already have an existing Eclipse deployment then you can only download the respective WSMO Studio plug-ins and copy them into your ECLIPSE/plugins directory. The following plug-ins are required (available for download at the WSMO Studio web site):

- Wsmo4j plug-in
- WSMO Studio runtime plug-in
- WSMO plug-in

\(^{22}\) Keep in mind that wsmo4j at present is based on Java 1.4.2, although it should run flawlessly on Java 1.5.x as well.

\(^{23}\) …assuming that all the requirements outlined in the previous sections are met.
6 APPENDIX B – TECHNICAL DOCUMENTATION

The technical documentation for WSMO Studio consists of:

- JavaDoc – detailed documentation of the java interfaces and classes comprising WSMO Studio
- Source files
- Extension point definitions – the Eclipse extension points of the respective plug-ins, that can be used from 3rd parties to plug new functionality or extend existing functionality
- Plug-in descriptors
- WSDL definition of the Repository interface

6.1 JavaDoc

The JavaDoc is available online at the WSMO Studio website24. At present the JavaDoc for plug-ins can be directly accessed at:


Note that the exact URLs may be changed in the future.

6.2 Source files

WSMO Studio sources are publicly available. Instructions on how to access the source repository (CVS) are available [http://www.wsmostudio.org/cvs-usage.html](http://www.wsmostudio.org/cvs-usage.html).

---

6.3 Plug-in Descriptors

6.3.1 Wsmo4j Plug-in

<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>
<plugin
    id="com.ontotext.wsmo4j"
    name="WSMO4J"
    version="0.4.0"
    provider-name="Ontotext Lab." class="com.ontotext.wsmo4j.Wsmo4j_Plugin">

    <runtime>
        <library name="wsmo4j-0.4.0.jar">
            <export name="*"/>
        </library>
        <library name="wsmlparser-20050601.jar">
            <export name="*"/>
        </library>
    </runtime>

    <requires>
        <import plugin="org.eclipse.ui"/>
        <import plugin="org.eclipse.core.runtime"/>
    </requires>

</plugin>

Listing 1 wsmo4j plug-in descriptor

6.3.2 WSMO Studio Runtime

<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>
<plugin
    id="org.wsmostudio.runtime"
    name="WSMO Studio Runtime Plugin"
    version="0.1.0"
    provider-name="Ontotext Lab." class="org.wsmostudio.runtime.RuntimePlugin"
<runtime>
  <library name="runtime.jar">
    <export name="*"/>
  </library>
</runtime>

<requires>
  <import plugin="org.eclipse.core.runtime"/>
  <import plugin="org.eclipse.ui"/>
  <import plugin="com.ontotext.wsmo4j"/>
  <import plugin="org.eclipse.core.resources"/>
</requires>

<extension
  point="org.eclipse.ui.startup">
  <startup class="org.wsmostudio.runtime.WsmoImageRegistry"/>
</extension>

</plugin>

Listing 2 WSMO Studio Runtime descriptor

6.3.3 WSMO Plug-in

<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>
<plugin
  id="org.wsmostudio.ui"
  name="WSMO Plug-in"
  version="0.1.0"
  provider-name="Ontotext Lab."
  class="org.wsmostudio.ui.WsmoUIPlugin">

<runtime>
  <library name="wsmostudio.jar">
    <export name="*"/>
  </library>
</runtime>

<requires>
  <import plugin="org.eclipse.ui"/>
  <import plugin="org.eclipse.core.runtime"/>
  <import plugin="org.eclipse.ui.views"/>
  <import plugin="org.eclipse.ui.ide"/>
</requires>
<import plugin="org.eclipse.core.resources"/>
<import plugin="com.ontotext.wsmo4j"/>
<import plugin="org.wsmostudio.runtime"/>
</requires>
<extension-point id="editors" name="WSMO Editors" schema="schema/editors.exsd"/>
<extension
  point="org.eclipse.ui.views">
  <view
    allowMultiple="false"
    class="org.wsmostudio.ui.views.navigator.WSMONavigator"
    icon="icons/navi.gif"
    category="org.wsmostudio.ui.views"
    name="WSMO Navigator"
    id="org.wsmostudio.ui.views.WSMOView"/>
  <category
    name="WSMO Studio"
    id="org.wsmostudio.ui.views"/>
</extension>
<extension
  point="org.eclipse.ui.editors">
  <editor
    icon="icons/concept.gif"
    class="org.wsmostudio.ui.editors.ConceptEditor"
    default="true"
    name="Concept Editor"
    id="org.wsmostudio.ui.editors.conceptEditor"/>
  <editor
    class="org.wsmostudio.ui.editors.RelationEditor"
    icon="icons/relation.gif"
    default="false"
    name="Relation Editor"
    id="org.wsmostudio.ui.editors.relationEditor"/>
  <editor
    icon="icons/instance.gif"
    class="org.wsmostudio.ui.editors.InstanceEditor"
    default="false"
    name="Instance Editor"
    id="org.wsmostudio.ui.editors.instanceEditor"/>
  <editor
    class="org.wsmostudio.ui.editors.OntologyEditor"
    icon="icons/ontoicon.gif"
    default="false"
    name="Ontology Editor"/>
<editor
    class="org.wsmostudio.ui.editors.ontologyEditor"
    icon="icons/capability.gif"
    default="false"
    name="Capability Editor"
    id="org.wsmostudio.ui.editors.capabilityEditor"/>

<editor
    class="org.wsmostudio.ui.editors.AxiomEditor"
    icon="icons/axiom.gif"
    default="false"
    name="Axiom Editor"
    id="org.wsmostudio.ui.editors.axiomEditor"/>

<editor
    class="org.wsmostudio.ui.editors.ServiceDescriptionEditor"
    icon="icons/webservice.gif"
    default="false"
    name="WebService Editor"
    id="org.wsmostudio.ui.editors.webserviceEditor"/>

<editor
    class="org.wsmostudio.ui.editors.InterfaceEditor"
    icon="icons/interface.gif"
    default="false"
    name="Interface Editor"
    id="org.wsmostudio.ui.editors.interfaceEditor"/>

<editor
    class="org.wsmostudio.ui.editors.InstanceEditor"
    icon="icons/instance.gif"
    default="false"
    name="Instance Editor"
    id="org.wsmostudio.ui.editors.instanceEditor"/>

<editor
    class="org.wsmostudio.ui.editors.RelationInstanceEditor"
    icon="icons/relinstance.gif"
    default="false"
    name="Relation Instance Editor"
    id="org.wsmostudio.ui.editors.relInstanceEditor"/>

<editor
    class="org.wsmostudio.ui.editors.MediatorEditor"
    icon="icons/wwmediator.gif"
    default="false"
    name="Mediator Editor"
    id="org.wsmostudio.ui.editors.mediatorEditor"/>
<editor
    launcher="org.wsmostudio.ui.editors.WSMOEditorLauncher"
    filenames="wsml"
    icon="icons/elunch.gif"
    default="true"
    name="WSMO Editor Launcher"
    id="org.wsmostudio.ui.editors.WSMOEditorLauncher"
    extensions="wsml"/>

<editor
    class="org.wsmostudio.ui.editors.ServiceDescriptionEditor"
    icon="icons/goal.gif"
    default="false"
    name="Goal Editor"
    id="org.wsmostudio.ui.editors.goalEditor"/>
</extension>

<extension
    point="org.eclipse.ui.perspectives">
    <perspective
        class="org.wsmostudio.ui.perspective.WSMOPerspecitveFactory"
        icon="icons/wsmo_small_up.gif"
        fixed="false"
        name="WSMO"
        id="org.wsmostudio.ui.perspective.wsmo"/>
</extension>

<extension
    point="org.eclipse.ui.newWizards">
    <category
        name="WSMO"
        id="org.wsmostudio.ui.newWizards"/>
    <wizard
        class="org.wsmostudio.ui.wizards.NewOntologyWizard"
        icon="icons/ontoicon.gif"
        category="org.wsmostudio.ui.newWizards"
        name="Ontology"
        id="org.wsmostudio.ui.wizards.newOntologyWizard">
        <description>Create a new ontology</description>
    </wizard>
    <wizard
        icon="icons/webservice.gif"
        class="org.wsmostudio.ui.wizards.NewWebServiceWizard"
        category="org.wsmostudio.ui.newWizards"
        name="Web Service"
        id="org.wsmostudio.ui.wizards.newWebService">
Create a new web service

Create a new goal

Create a new mediator

Create a new WSMO project

editorId="org.wsmostudio.ui.editors.axiomEditor"

wsmoType="org.omwg.ontology.Axiom"/?>
Listing 3 WSMO plug-in descriptor

6.3.4 Repository plug-in

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>
<plugin
id="org.wsmostudio.repository"
name="Repository Plugin"
version="0.1.0"
provider-name="Ontotext Lab."
class="org.wsmostudio.repository.RepositoryPlugin">

<runtime>
  <library name="repository.jar">
    <export name="*"/>
  </library>
</runtime>

<requires>
  <import plugin="org.eclipse.ui"/>
  <import plugin="org.eclipse.core.runtime"/>
  <import plugin="org.wsmostudio.runtime"/>
  <import plugin="org.eclipse.ui.ide"/>
  <import plugin="org.eclipse.core.resources"/>
  <import plugin="com.ontotext.wsmo4j" export="false"/>
</requires>

<extension-point id="Repository" name="Repository Extension"
schema="schema/org.wsmostudio.repository.exsd"/>
<extension
  point="org.eclipse.ui.views">
  <view
    icon="icons/repo_r.gif"
    class="org.wsmostudio.repository.ui.RepositoriesExplorer"
    category="org.wsmostudio.ui.views"
```
6.4 Eclipse Extension Points

This section presents technical details on the extension points published by the WSMO Studio plug-ins.

We highly recommend that one always obtains the most recent information on an extension point from the web site (available in the plug-in descriptor and extension schemas) and does not rely exclusively on the information in this section, since it may be outdated.

6.4.1 Repository extensions

The Repository plug-in defines a single extension point called “Repository” (see the plug-in descriptor in section 6.3.4).
The repository extension allows that a new repository façade is plugged-in the repository plug-in. The façade should implement the `org.wsmo.datastore.WsmoRepository` interface from section 3.4.

In addition a class that implements the `org.wsmostudio.runtime.extension.Configurator` interface should be provided so that the remote repository can be initialised (e.g. URL, user name, password, etc.)


### 6.4.2 WSMO Editors

The WSMO plug-in defines an extension that allows various editors (such as Axiom editors, Interface editors, etc.) to be replaced by 3rd party implementations.

An editor is a visual component within a workbench page which is typically used to edit or browse a wsmo object. When an object definition must be opened, the studio's registry is consulted to determine an appropriate editor for the object type and then a new instance of the editor type is created.

As *WSMO Studio* is strongly tied with the Eclipse Platform, the specific WSMO editors are also Eclipse editors. Furthermore, each new WSMO editor must extend Eclipse's extension point `org.eclipse.ui.editors`. Additionally, bindings between WSMO objects and corresponding editors are needed. Each such binding registers an editor which will be used when a certain WSMO object is opened.

#### 6.4.2.1 Extension point schema

```xml
<!ELEMENT extension (editorBinding*)>
<!ATTLIST extension
  point CDATA #REQUIRED
  id   CDATA #IMPLIED
  name CDATA #IMPLIED>
  • point - a fully qualified identifier of the target extension point
  • id - an optional identifier of the extension instance
  • name - an optional name of the extension instance

<!ELEMENT editorBinding EMPTY>
<!ATTLIST editorBinding
  editorId CDATA #REQUIRED
  wsmoType CDATA #REQUIRED
  default (true | false) "false">
  • editorId - an existing editor extension reference for extension point `org.eclipse.ui.editors`.
  • wsmoType - a fully qualified class name of a WSMO object which the referenced editor can handle as input.
  • default - if true, this editor will be used as the default editor for the type. This is only relevant in a case where more than one editor is registered for the same type.
```
6.4.2.2 Example

The following is an example of a WebService editor extension definition (here we assume that 'org.wsmostudio.ui.editors.ServiceDescriptionEditor' is already defined as Eclipse editor extension):

```xml
<extension point="org.wsmostudio.ui.editors">
  <editorBinding
    editorId = "org.wsmostudio.ui.editors.ServiceDescriptionEditor"
    wsmoType = "org.wsmo.service.WebService"
    default = "true">
  </editor>
</extension>
```

6.4.2.3 API Information

In the extension definition of `org.eclipse.ui.editors`, the class attribute specifies a class which must implement `org.eclipse.ui.IEditorPart`. The implementing class may extend class `org.wsmostudio.ui.WSMOEditor` (recommended) which supplies an implementation of `IEditorPart` with some additional facilities.

The mechanism for retrieving the WSMO object from the `IEditorInput` (passed to the editor implementation) uses the `IAdaptable` interface (which `IEditorInput` extends).

The following snippet demonstrates the technique:

```java
/**
 * implements IEditorPart.init()
 */
public void init(IEditorSite site, IEditorInput input) throws PartInitException {
  Ontology inputOntology = (Ontology)input.getAdapter(Ontology.class);
}
```

If the implementation requests an object of incorrect type - `null` is returned.

6.5 Repository Interface

The WSDL definition of the Repository is available here: [http://www.wsmostudio.org/doc/repository.wsdl](http://www.wsmostudio.org/doc/repository.wsdl)

7 Appendix C – Licence
7.1 WSMO Studio Licence

WSMO Studio is provided under a LGPL licence\(^{25}\). Note that 3\(^{rd}\) party extensions or new plug-ins may be provided under a proprietary licence chosen by the respective contributor\(^{26}\), in which case the contributing party should explicitly indicate the detailed licensing terms for its contribution.

7.2 3\(^{rd}\) Party Dependencies and Licences

Table 3 shows the 3\(^{rd}\) party modules or libraries part of the WSMO Studio distribution, together with their respective licences.

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
<th>Licence</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse</td>
<td>3.1.0</td>
<td>Eclipse Public Licence(^{27})</td>
<td><a href="http://www.eclipse.org/">http://www.eclipse.org/</a></td>
</tr>
<tr>
<td>Axis</td>
<td>1.2</td>
<td>Apache Software Licence(^{28})</td>
<td><a href="http://ws.apache.org/axis/">http://ws.apache.org/axis/</a></td>
</tr>
<tr>
<td>wsmo4j</td>
<td>0.4.0</td>
<td>LGPL(^{29})</td>
<td><a href="http://wsmo4j.sourceforge.net">http://wsmo4j.sourceforge.net</a></td>
</tr>
</tbody>
</table>

\(^{25}\) [http://www.opensource.org/licenses/lgpl-license.php](http://www.opensource.org/licenses/lgpl-license.php)

\(^{26}\) LGPL allows that a software component or a library licensed under LGPL (open source) is bundled together with proprietary, non-open source components.


\(^{28}\) [http://ws.apache.org/LICENSE.txt](http://ws.apache.org/LICENSE.txt)

\(^{29}\) [http://www.opensource.org/licenses/lgpl-license.php](http://www.opensource.org/licenses/lgpl-license.php)