



D9.1v0.1 Web Service Modeling Toolkit (WSMT)

WSMX Working Draft 27 January 2005

This version:

<http://www.wsmo.org/TR/d9/d9.1/v0.1/20050127>

Latest version:

<http://www.wsmo.org/TR/d9/d9.1/v0.1/>

Previous version:

<http://www.wsmo.org/TR/d9/d9.1/v0.1/>

Editors:

Mick Kerrigan

Authors:

Mick Kerrigan

Reviewers:

Adrian Mocan

This document is also available in non-normative [PDF](#) version.

Copyright © 2005 [DERI](#)®, All Rights Reserved. [DERI](#) liability, trademark, document use, and software licensing rules apply.

Table of contents

1. Introduction

[1.1 Overview](#)

[1.2 Purpose of this Document](#)

[1.3 Downloading the Web Services Modeling Toolkit](#)

2. Web Services Modeling Toolkit

3. Creating a Tool as a Plugin

4. Future Work

References

Acknowledgement

1. Introduction

1.1 Overview

The Web Service Modeling Ontology (WSMO) [[Roman et al., 2004](#)] is an ontology for describing semantic web services. WSMO is based on the Web Service Modeling Framework (WSMF) [[Fensel & Bussler, 2002](#)] and as such is based on the four main elements of the WSMF: Ontologies, Goals, Mediators and Web Services. The Web Service Modeling Language (WSML) [[de Bruijn et al., 2004](#)] is a formalization of the WSMO ontology and provides a number of languages within which the properties of semantic web services can

be described. These languages are based on Description Logic and Logic Programming, each language variant providing "different levels of logical expressiveness" [[de Bruijn et al., 2004](#)]. The Web Service Execution Environment (WSMX) is a reference implementation of WSMO using the WSML family of languages. WSMX is an execution environment for the dynamic discovery, mediation, composition and invocation of semantic web services. WSMX uses WSMO as its conceptual model and defines its own execution semantics [[Oren, 2004](#)], architecture [[Zaremba et al., 2004](#)] and implementation [[Moran, 2004](#)].

1.2 Purpose of this Document

The purpose of this document is to outline the Web Services Modeling Toolkit (WSMT). WSMT is a framework for the rapid deployment of graphical administrative tools, which can be used with WSMO, WSML and WSMX.

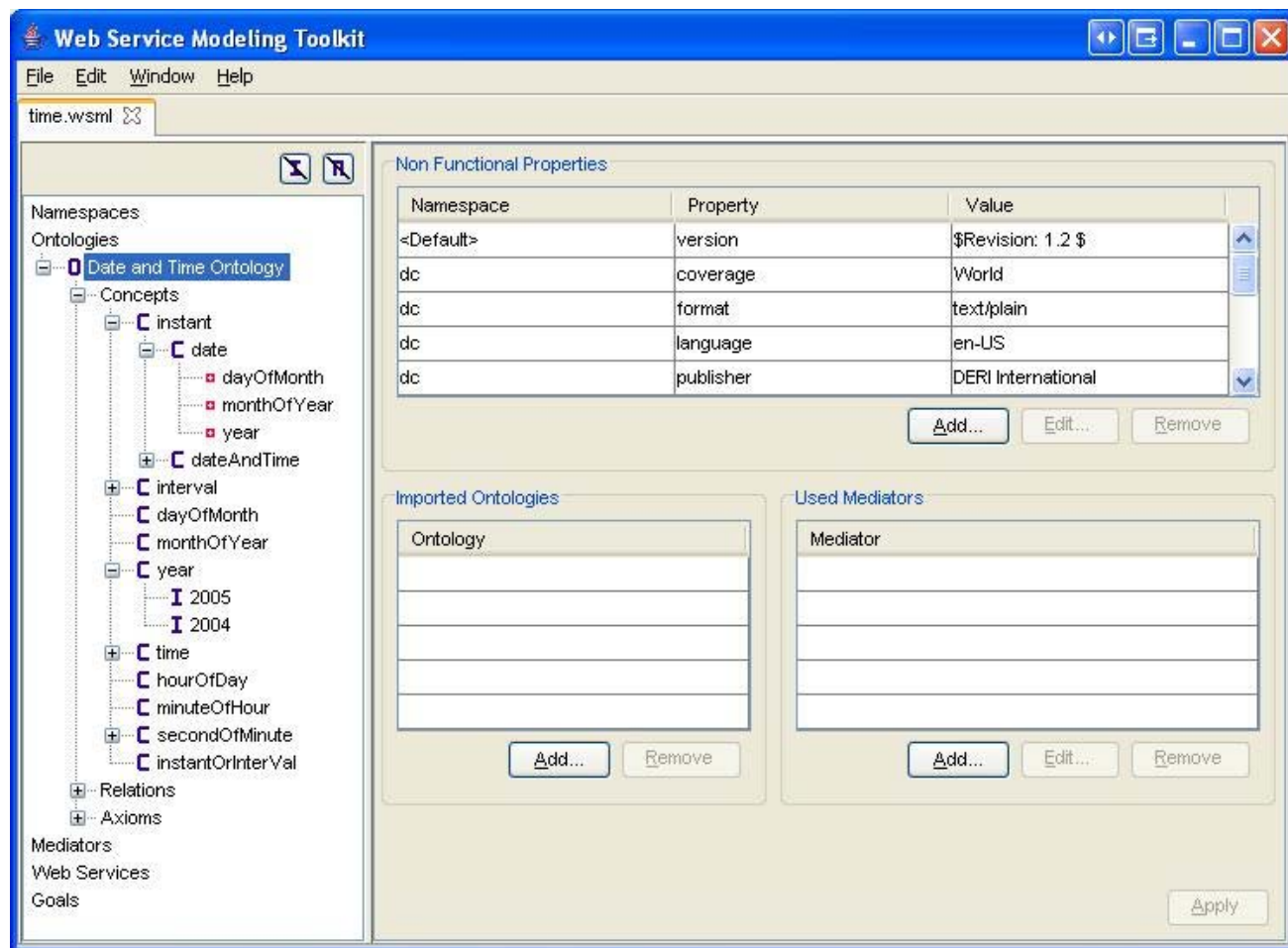
1.3 Downloading the Web Services Modeling Toolkit

The latest version (vMajor.Minor) of the WSMT is available [here](#).

2. Web Services Modeling Toolkit

As research into the semantic web continues within the WSMO, WSML and WSMX clusters at DERI, the need for tools to manage resources and systems is becoming more apparent. The WSMT provides the functionality to deploy these tools as plugins to a larger system, allowing the users of semantic web technology to install one client side application for multiple tasks. For tool developers the framework provides the basic functionality for housing a tool and reduces the need for each developer to redevelop this functionality for each tool. This enables increased developer productivity and allows a developer to rapidly develop tools for WSMO and WSML or for testing the components in the WSMX system.

The WSMT currently provides a windows install, complete with Java 1.5 runtime, dynamic classpath generation, plugin detection and localization framework. Two tools to be created within the WSMT have been planned, the WSML Editor and the WSMX Monitor. These tools will be discussed in further detail in deliverables d9.2 [Kerrigan, 2005a] and d9.3 [Kerrigan, 2005b] respectively.



3. Creating a Tool as a Plugin

The process for creating a tool as a plugin is a very straight forward one. The source code for the tool should be packed in a jar and placed in the lib folder of the WSMT installation along with the plugin description file (.pdesc). WSMT reads all the plugin description files at startup and loads each of the plugins specified. This list of installed plugins can be found in the plugins section for the about dialog.

```
<?xml version="1.0" encoding="UTF-8"?>
<pluginDescription version="1.0">
```

```

    <id>wsmleditor</id>
    <version>0.1</version>
    <classname>ie.deri.wsmtool.wsmleditor.WSMLEditorPlugin</classname>
</pluginDescription>

```

wsmleditor.pdesc

The plugin description file allows WSMT to perform plugin detection. The file gives the plugin's id, version and the classname for creating an instance of the tool. The classname specified is that of an implementation of the `ie.deri.wsmtool.plugin.Plugin` interface.

```

package ie.deri.wsmtool.plugin;

import ie.deri.wsmtool.gui.PluginPanel;

import java.awt.Container;
import java.awt.Dimension;
import java.util.Locale;

public interface Plugin {

    public void init(String theInstallPath, PluginDescription
thePluginDescription, Locale theLocale, Container theParent) throws
PluginInitializationFailedException;
    public String getName();
    public String getLabel();
    public String getVersion();
    public String getDescription();
    public PluginPanel getPanel();
    public Dimension getDimension();
}

```

The tool itself is created by creating a subclass of the abstract `PluginPanel` class, which is a subclass of `JPanel`.

```

package ie.deri.wsmtool.gui;

import javax.swing.JMenuBar;
import javax.swing.JPanel;

public abstract class PluginPanel extends JPanel{

    public abstract JMenuBar getMenuBar(ActionListener refreshMenuBar);
}

```

4. Future Work

A toolbar will be added to the WSMT to which plugins can contribute items. This will be done by adding an additional method to the `pluginpanel` class called `getToolBar()`.

Currently Plugins are displayed in the main panel of the applications as tabs. It should be possible for the user to choose which plugins (if any) are displayed in the main view of the application. A mechanism for showing, hiding and switching between different plugins needs to be devised.

The Plugin description format is very simple at the moment, some additional values will be added to give added value to the application. A new `author` tag can be used to specify 1 or more authors of the plugin. Two new tags for compatibility reasons, `WSMT-version` and `java-version` will be added so that plugin compatibility functionality can be added to the application.

Currently each plugin is responsible for finding the files which it needs, relative to the installation path of the application and the plugin name. To reduce this overhead a resource finding component will be added to the WSMT core.

Each of the plugins is also currently responsible for implementing its own error logging. In the future this

should be replaced with a logging component in the WSMT core. The component will use Log4J or a similar library to perform logging, therefore the end-user will be able to configure the logging of the application through one configuration file.

As the need for communication with the WSMX system is required by plugins to the WSMT, a generic WSMX communication component, which can be used any of the plugins to communicate with the WSMX system will be added to the WSMT core. As the functionality in the WSMX system is improved, this component will be changed to be a WSMX using component, i.e. all WSMX services needed by the WSMT will be semantically described and the WSMT will use WSMX to discover and invoke them.

As windows is so widely used it, it was chosen as the initial delivery platform for the WSMT. However as the WSMT is developed in the Java programming language it is as operating system independent as Java. Future work will include creation of install applications for Unix and Mac.

As the need for further tools are required they can be developed to work within the WSMT. Already discussions regarding WSMX Management, WSMX Mediation and WSMX Choreography tools have begun.

References

[de Bruijn et al., 2004] J. de Bruijn, H. Lausen, and D. Fensel, The WSML Family of Representation Languages, <http://www.wsmo.org/2004/d16/d16.1/v0.2/>

[Fensel & Bussler, 2002] D. Fensel and C. Bussler: The Web Service Modeling Framework WSMF, Electronic Commerce Research and Applications, 1(2), 2002.

[Kerrigan, 2005a] M. Kerrigan, WSML Editor, <http://www.wsmo.org/2005/d9/d9.2/v0.1/>

[Kerrigan, 2005b] M. Kerrigan, WSMX Monitor, <http://www.wsmo.org/2005/d9/d9.3/v0.1/>

[Moran, 2004] M. Moran, WSMX Implementation, <http://www.wsmo.org/2004/d13/d13.5/v0.1/>

[Oren, 2004] E. Oren. WSMX Execution Semantics, <http://www.wsmo.org/2004/d13/d13.2/v0.1/>

[Roman et al., 2004] D. Roman, H. Lausen, U. Keller (eds.): Web Service Modeling Ontology (WSMO), <http://www.wsmo.org/2004/d2/v1.0/>

[Zaremba et al., 2004] M. Zaremba, M. Moran, WSMX Architecture, <http://www.wsmo.org/2004/d13/d13.4/v0.1/>

Acknowledgement

This work is funded by the European Commission under the projects DIP, Knowledge Web, Ontoweb, SEKT, SWWS, Esperonto, and h-TechSight; by Science Foundation Ireland under the DERI-Lion project; and by the Vienna city government under the CoOperate programme.

The editors would like to thank to all the members of the WSMX working group for their advises and inputs to this document.



[webmaster](#)