DIP
Data, Information and Process Integration with Semantic Web Services

FP6 - 507483

Deliverable

WP 6: Interoperability and Architecture

D6.7

Architecture Prototype V1

Thomas Haselwanter
Bernhard Schreder
Alexander Wahler
Michal Zaremba

July 10th, 2004
EXECUTIVE SUMMARY

This deliverable provides an overview of the first version of the DIP architecture prototype including detailed installation guidelines and documentation. The prototype itself consists of the WSMX core architecture as well as several additional components.

WSMX [8] is an execution environment which enables discovery, selection, mediation, invocation and interoperation of the Semantic Web Services (SWS). The mission and ultimate goal of the WSMX working group is to define a SWS architecture and build a fully fledged enterprise application based on the conceptual model of WSMO [7]. WSMX is based on the conceptual model provided by WSMO, being at the same time a reference implementation of it. It is the scope of WSMX to provide a test bed for WSMO and to prove its viability as a mean to achieve dynamic interoperability of Semantic Web Services. In this document we report on the open source implementation of the system, which is used as a reference implementation of the DIP architecture.

Since July 2004, the code base of WSMX is hosted at SourceForge – the world’s largest repository of open source projects. The WSMX open source implementation can be accessed at http://sourceforge.net/projects/wsmx/, where both current and previous releases, as well as all the code are available.

An instance of the prototype can also be accessed online at http://wsmx.deri.org:7000/ and the visualisation GUI of this instance can be found at http://62.116.8.109:8080/visualisation/.

This deliverable contributes to the Open Source Semantic Web Service Architecture and is relevant to all components developed in DIP. The target audiences of this deliverable are developers and IT experts who are interested to test and evaluate the prototype.

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.
Document Information

<table>
<thead>
<tr>
<th>IST Project Number</th>
<th>FP6 – 507483</th>
<th>Acronym</th>
<th>DIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full title</td>
<td>Data, Information, and Process Integration with Semantic Web Services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project URL</td>
<td><a href="http://dip.semanticweb.org">http://dip.semanticweb.org</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document URL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EU Project officer</td>
<td>Kai Tullius</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Deliverable Number | 6.7 | Title | Architecture Prototype V1 |
Work package Number | 6 | Title | Interoperability and Architecture |

Date of delivery | Contractual M18 | Actual 10-Jul-04 |
Status | version. 0.01 final |
Nature | Prototype Report Dissemination |
Dissemination Level | Public Consortium |

Authors (Partner) | Thomas Haselwanter (UIBK), Bernhard Schreder (NIWA), Alexander Wahler (NIWA), Michal Zaremba (NUIG) |
Responsible Author | Bernhard Schreder Email schreder@niwa.at |
| Partner | NIWA Phone +4313195843-13 |

Abstract (for dissemination) | This deliverables provides an overview of the first version of the DIP architecture prototype including detailed installation guidelines and documentation. |
Keywords | Execution environment, architecture, semantic web, semantic web services, execution semantics, prototype |

Version Log

<table>
<thead>
<tr>
<th>Issue Date</th>
<th>Rev No.</th>
<th>Author</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-jul-05</td>
<td>001</td>
<td>Bernhard Schreder</td>
<td>Initial Version</td>
</tr>
<tr>
<td>07-jul-05</td>
<td>002</td>
<td>Alexander Wahler</td>
<td>Executive Summary, Conclusion added</td>
</tr>
<tr>
<td>10-jul-05</td>
<td>003</td>
<td>Bernhard Schreder</td>
<td>Updates to section 2 and 3</td>
</tr>
<tr>
<td>Date</td>
<td>Code</td>
<td>Name</td>
<td>Comments</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>-------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>18-jul-05</td>
<td>004</td>
<td>Bernhard Schreder</td>
<td>Integration of reviewers comments</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.nj.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 Rowlatt,  
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: abercker@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: j.b.domingue@open.ac.uk  
Tel.: +44 1908 655014 |
| SAP AG | SAP | Dr. Elmar Dörner  
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 Person(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sirma AI Ltd.</td>
<td>Atanas Kiryakov, Ontotext Lab, - Sirma AI EAD</td>
</tr>
<tr>
<td></td>
<td>Office Express IT Centre, 3rd Floor</td>
</tr>
<tr>
<td></td>
<td>135 Tzarigradsko Chausse</td>
</tr>
<tr>
<td></td>
<td>Sofia 1784, Bulgaria</td>
</tr>
<tr>
<td></td>
<td>Email: <a href="mailto:atanas.kiryakov@sirma.bg">atanas.kiryakov@sirma.bg</a></td>
</tr>
<tr>
<td></td>
<td>Tel.: +359 2 9768 303</td>
</tr>
<tr>
<td>Unicorn Solution Ltd.</td>
<td>Jeff Eisenberg, Unicorn Solutions Ltd.</td>
</tr>
<tr>
<td></td>
<td>Malcha Technology Park 1</td>
</tr>
<tr>
<td></td>
<td>Jerusalem 96951, Israel</td>
</tr>
<tr>
<td></td>
<td>Email: <a href="mailto:Jeff.Eisenberg@unicorn.com">Jeff.Eisenberg@unicorn.com</a></td>
</tr>
<tr>
<td></td>
<td>Tel.: +972 2 6491111</td>
</tr>
<tr>
<td>Vrije Universiteit Brussel</td>
<td>Pieter De Leenheer, Starlab- VUB</td>
</tr>
<tr>
<td></td>
<td>Vrije Universiteit Brussel</td>
</tr>
<tr>
<td></td>
<td>Pleinlaan 2, G-10, 1050 Brussel, Belgium</td>
</tr>
<tr>
<td></td>
<td>Email: <a href="mailto:Pieter_De_Leenheer@vub.ac.be">Pieter_De_Leenheer@vub.ac.be</a></td>
</tr>
<tr>
<td></td>
<td>Tel.: +32 (0) 2 629 3749</td>
</tr>
</tbody>
</table>
LIST OF KEY WORDS/ABBREVIATIONS

GUI  graphical user interface
IDE  Integrated Development Environment
JINI  a network architecture for the construction of distributed systems
SOA  Service Oriented Architecture
SWS  Semantic Web Services
TUI  text user interface
WSMO  Web Services Modelling Ontology
WSMX  Web Services Execution Environment
# TABLE OF CONTENTS

EXECUTIVE SUMMARY ........................................................................................................... 1
LIST OF KEY WORDS/ABBREVIATIONS ............................................................................... VII
TABLE OF CONTENTS ............................................................................................................. VIII
1 INTRODUCTION .................................................................................................................. 1
2 ARCHITECTURE PROTOTYPE FACT SHEET ..................................................................... 1
   2.1 Description of purpose, scope and functionality ......................................................... 1
   2.2 Available Release and Components ............................................................................. 2
   2.3 Installation Guidelines ................................................................................................. 3
       2.3.1 WSMX Software Prerequisites ......................................................................... 3
       2.3.2 Inca X installation and usage instructions ......................................................... 4
       2.3.3 WSMX Installation Instructions ....................................................................... 4
       2.3.4 Visualisation of WSMX Execution ................................................................. 5
   2.4 Type of API .................................................................................................................. 6
   2.5 Licence information ...................................................................................................... 6
   2.6 How to use the prototype ............................................................................................ 6
   2.7 Roadmap for future plans ............................................................................................ 6
3 ARCHITECTURE PROTOTYPE DOCUMENTATION ............................................................ 7
   3.1 Server Administration GUI ....................................................................................... 7
   3.2 Visualisation GUI ....................................................................................................... 9
   3.3 Additional documentation .......................................................................................... 10
4 CONCLUSION ........................................................................................................................ 10
REFERENCES ........................................................................................................................... 11
APPENDIX 1: WSMX VISUALISATION ................................................................................. 12
APPENDIX 2: THIRD PARTY SOFTWARE .............................................................................. 13

# LIST OF FIGURES

Figure 1: future WSMX architecture .................................................................................... 7
Figure 2: WSMX Management Console – Main View ......................................................... 7
Figure 3: WSMX Management Console – Server View ...................................................... 8
Figure 4: WSMX Management Console – Mbean View (CommunicationManager) ....... 9
Figure 5: WSMX Visualisation .......................................................................................... 10
LIST OF TABLES

Table 1: WSMX packages ................................................................. 2
Table 2: mockup WSMX components ......................................... 3
Table 3: real WSMX components ................................................... 3
Table 4: Third Party Software and Licences ...................................... 13
1 INTRODUCTION

This deliverable provides the documentation for the first version of the DIP architecture prototype, representing an implementation of the architecture first specified in [1], respectively in the revised architecture deliverable [4]. The prototype itself consists of the WSMX core architecture as well as several additional components. As a prototype deliverable, this document is structured as follows: Section 2 consists of the Architecture Prototype Fact Sheet, which contains all the relevant information concerning the prototype, i.e. installation guidelines. Section 3 contains a description of the GUIs, as well as the links to additional documentation on used APIs. Finally, the conclusion summarises the status of the prototype architecture and briefly explains how the prototype can be demonstrated.

2 ARCHITECTURE PROTOTYPE FACT SHEET

The contact person for the architecture prototype is Michal Zaremba (michal.zaremba@deri.org)

2.1 Description of purpose, scope and functionality

WSMX is an execution environment which enables discovery, selection, mediation, invocation and interoperation of the Semantic Web Services (SWS). The mission and ultimate goal of the WSMX working group¹ is to define a SWS architecture and build a fully fledged enterprise application based on the conceptual model of WSMO. WSMX is based on the conceptual model provided by WSMO, being at the same time a reference implementation of it. It is the scope of WSMX to provide a test bed for WSMO and to prove its viability as a mean to achieve dynamic interoperability of Semantic Web Services. In this document we report on the open source implementation of the system.

WSMX functionalities can be classified in two main categories: first, the functionality that should be part of any environment for Semantic Web Services and second, the additional functionality coming from the enterprise system features of the framework. In the first case, the overall WSMX functionality can be seen as an aggregation of the components' functionalities, which are part of the WSMX architecture. In the second case, WSMX offers features such as a plugging mechanism that allows the integration of various distributed components, an internal workflow engine capable of executing formal descriptions of the components behaviour or a resource manager that enables the persistency of WSMO and non-WSMO data produced during run-time. The main components that have already been designed and have been implemented or are under implementation in WSMX are: Core Component, Resource Manager, Discovery, Selection, Data and Process Mediator, Communication Manager, Choreography Engine, Web service Modelling Toolkit and Reasoner.

The WSMX architecture consists of a set of loosely coupled components. Having various loosely coupled components as part of a software system is one of the fundamental principles of a Service Oriented Architecture (SOA). (principles of Service Oriented Architectures are detailed for example in [9]) Self-contained components with

¹ The working group’s relation to DIP is explained in [11].
well defined functionalities can be easily plugged-in and removed at any time, allowing them to use each others functionalities. Even if WSMX provides a default implementation for all the components in the architecture, following these principles allows third-party components offering the same functionality (or an enhanced functionality) to be easily plugged-in.

The WSMX architecture aims to provide descriptions of the external interface and behaviour for all the components and for the system as a whole. By this, the system’s overall functionality is separated from the implementation of particular components.

Since July 2004, the code base of WSMX is hosted at SourceForge – the world’s largest repository of open source projects. The WSMX open source implementation can be accessed at http://sourceforge.net/projects/wsmx/, where both current and previous releases, as well as all the code are available.

In addition to the mailing list of the WSMX working group, three mailing lists have been established to facilitate communication for WSMX developers and users.²

2.2 Available Release and Components

The current release of WSMX (version 0.2) is available in the form of different packages. A description of the packages and the included components and libraries is shown in Table 1:

<table>
<thead>
<tr>
<th>Package name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsmx_lite-0.2-src.zip</td>
<td>sources without third party libraries</td>
</tr>
<tr>
<td>wsmx_all-0.2-src.zip</td>
<td>sources with third party libraries</td>
</tr>
<tr>
<td>wsmx-thirdparty-0.2.zip</td>
<td>only third party libraries</td>
</tr>
<tr>
<td>wsmx_core_and_mockups-0.2-bin.zip</td>
<td>compiled version of core engine and mockup components</td>
</tr>
<tr>
<td>wsmx_core_with_branch_of_execution_semantics-0.2-bin.zip</td>
<td>compiled version of core engine with one working branch of execution semantics</td>
</tr>
<tr>
<td>wsmx_components-0.2-bin.zip</td>
<td>compiled components</td>
</tr>
<tr>
<td>wsmx-integration-API-0.2.1b.zip</td>
<td>Contains the Integration API and corresponding javadoc</td>
</tr>
</tbody>
</table>

There is also an instance of WSMX running on a server of the University of Innsbruck at http://wsmx.deri.org:7000/

The packages include a number of mockup components which will be injected into the WSMX Kernel upon startup, as shown in Table 2.

² see http://sourceforge.net/mail/?group_id=113321 for details
Table 2: mockup WSMX components

<table>
<thead>
<tr>
<th>Component name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChoreographyEngine</td>
</tr>
<tr>
<td>CommunicationManager</td>
</tr>
<tr>
<td>DataMediator</td>
</tr>
<tr>
<td>Discovery</td>
</tr>
<tr>
<td>Negotiator</td>
</tr>
<tr>
<td>Selector</td>
</tr>
<tr>
<td>Parser</td>
</tr>
<tr>
<td>ProcessMediator</td>
</tr>
<tr>
<td>ResourceManager</td>
</tr>
</tbody>
</table>

In addition to the mockup components a number of real components are included in this release of WSMX. The package wsmx_components-0.2-bin includes the following components with the described functionality. A detailed description of these components can be found in [4].

Table 3: real WSMX components

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyword_discovery</td>
<td>Keyword based discovery</td>
</tr>
<tr>
<td>pellet_discovery</td>
<td>Pellet based discovery</td>
</tr>
<tr>
<td>AlphaParser</td>
<td>The Parser checks if the syntax of received WSML descriptions is correct.</td>
</tr>
<tr>
<td>FZIReasoner</td>
<td>Prototype version of WSMO Reasoner</td>
</tr>
<tr>
<td>RuntimeMediator</td>
<td>Two versions of the run-time data mediation are provided: stand-alone and as a web service</td>
</tr>
</tbody>
</table>

2.3 Installation Guidelines

2.3.1 WSMX Software Prerequisites

The following list specifies the needed third party products and required libraries to use the prototype:

- JDK 5.0 (Download from http://java.sun.com/j2se/1.5.0/download.jsp)
- JINI - includes JavaSpaces implementation. Download the Jini Technology Starter Kit (v2.0.x) from http://starterkit.jini.org/downloads/. Extract the
downloaded file to a directory of your choice. JINI and the JavaSpaces technology is detailed in the JINI Specifications and API Archive\(^3\).

- Inca X - facilities starting JavaSpaces. Download Inca X & JavaSpaces Starter Kit v3.5 from http://www.incax.com/, follow the installation and usage guidelines below.

### 2.3.2 Inca X installation and usage instructions

The current WSMX release relies on a JavaSpaces compliant space implementation. Common examples include Outrigger, the implementation by Sun, and Blitz\(^4\), an open source implementation.

Inca X is a visual development and deployment platform for JINI technology. Run the installer program from your local disk drive by using the full path to your JDK 5.0 as illustrated below.

**Unix:** `/usr/java/j2sdk1.4.x/bin/java -jar install-incax.jar`

**Windows:** `c:\j2sdk1.4.x\bin\java -jar install-incax.jar`

The installer asks for the path to the JINI download (see above).

For the purposes of starting and using the supplied WSMX prototype a transient JavaSpace is sufficient. Run Blitz transient by starting the incaX IDE and switch from the “Development” to the “Runtime” tab. Wait for services to appear in the service directory and then expand the “Blitz JavaSpace” node. Right-click on the “Blitz [transient]” node and select “Start Blitz [transient]”. For further details you can also start its GUI console to see statistics of entries, blocking operations, etc. Click on “Run Service UI” in the Service UIs window, then select the “Factories->JFrameFactory” menu in the pop-up window to open the Blitz dashboard.

### 2.3.3 WSMX Installation Instructions

The kernel configuration is responsible for several configuration aspects of an instance of the WSMX microkernel, like the location of the systemcodebase, ports for the HTTP management console and SSH management console, or information on the used space. It might also include a list of domains, which in turn contains a list of components, for purposes of including components that are outside of the systemcodebase.

The systemcodebase is a location on the filesystem where component implementations reside and that is defined in the WSMX Kernel Configuration. Usually this is a directory populated with WSMX archives. WSMX scans the systemcodebase at startup and continues monitoring it to pick up components that are to be hot-deployed. Since everything that doesn't have a .wsmx extension is ignored, the WSMX executable jar, configuration, key and policy files are also found in the systemcodebase.

Components located in the systemcodebase don't need to be included in the kernel configuration since they will individually and automatically be scanned for a component.

---


\(^4\) [http://www.dancres.org/blitz/index.html](http://www.dancres.org/blitz/index.html)
configuration. Listing 1 shows a sample configuration file, after which a step-by-step instruction for setting up the wsmx_core_and_mockups-0.2-bin package is given.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<k:kernel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:k="http://www.wsmx.org/kernelconfiguration"
  systemcodebase="/path/to/codebase"
  httpport="8080"
  sshport="22"></k:kernel>
```

**Listing 1: WSMX kernel configuration**

- Edit the systemcodebase path in dist/config.xml to suit your environment. It should point to the directory with the WSMXcore.jar and the deployed wsmx components. (in this release this is the “dist” directory)

- Start up a JavaSpaces compliant space implementation such as Outrigger or Blitz. An easy way to set up a transient JavaSpace is detailed under the Inca X section above.

- Use Ant with the supplied buildfile, the default target “run” starts WSMX.

- The executable jar in the “dist” directory can also be run from the command line with sufficient privileges granted. A sample policy file (policy.all) which grants unrestricted access is supplied with the release.

  ```
  java -Djava.security.policy=/path/to/policy -jar WSMXcore.jar
  ```

- To deploy a component, package it in a WSMX archive and copy it to the systemcodebase. WSMX will discover it automatically and inject it into the running instance. A WSMX archive is a jar with a .wsmx extension and an agreed upon internal structure. The class files that make up the components implementations go to /classes, the archives deployment descriptor (if any) goes to /META-INF, libraries go jarred to /lib. WSMX uses custom classloaders that extract embedded jars, resolve load requests to the individual libraries, and provide isolation domains which allow components to load different version of the same class.

- You may monitor and administer WSMX through either the GUI-based HTTP management console or the TUI-based SSH management console.

2.3.4 Visualisation of WSMX Execution

The release of the first prototype includes a web application and a web service, in order to provide for a basic visualisation of a running WSMX instance. To set up this visualisation web application together with the web service which receives WSMX status information, follow the guidelines in Appendix 1.
For the online version of the visualisation GUI, which receives status information from the instance of WSMX running on the UIBK server, point your browser to the following URL: http://62.116.8.109:8080/visualisation

The visualisation itself is further detailed in section 3.2.

2.4 Type of API

The APIs for components interfaces (their sources, binaries and documentation) is available for download at: http://sourceforge.net/projects/wsmx. Third party component providers should download the newest version of the WSMX Integration API, in order for their components to be compatible with the WSMX architecture.

Additional APIs used for the architecture prototype include the WSMO API [2].

2.5 Licence information

WSMX uses the GNU General Public Licence\(^5\).

The third party software components and libraries included in the current WSMX release are using diverse licences, as shown in Appendix 2.

More detailed information about licensing of DIP components are provided in [5].

2.6 How to use the prototype

Following the steps detailed in section 2.3.3, WSMX should be running on a local machine. The WSMX Management Console can then be accessed by pointing your browser to http://localhost:port, where port is the port number which has been defined in the kernel configuration (config.xml) of the WSMX package.

The GUI documentation in section 3.1 details the possibilities of managing WSMX with the management console, including details on starting and stopping the server, as well as selecting and manipulating components.

2.7 Roadmap for future plans

The WSMX working group aims to continue its work on the WSMX system development. Several mockup components are going to be substituted with their real implementations. Also new strategic directions for the platform might be undertaken after the conclusion of the DIP project, such as orienting the platform towards the Semantic Grid. A future version for the WSMX architecture is presented in Figure 1.

\(^5\) http://www.opensource.org/licenses/gpl-license.php
3 ARCHITECTURE PROTOTYPE DOCUMENTATION

3.1 Server Administration GUI

The WSMX Management Console facilitates basic server administration tasks. Different views allow for the starting and stopping of the server (the main view, as seen in Figure 2), the administration of the deployed server components (the server view, as seen in Figure 3). Selecting one of the components from the server view opens up the MBean view of this service.

![Figure 1: future WSMX architecture](image)

![Figure 2: WSMX Management Console – Main View](image)
Figure 4 shows the MBean view of the CommunicationManager component, which enables a user to send a message to WSMX, using the “demo” operation of the CommunicationManager. This invokes the instantiation of the execution semantics (as defined in [10]) for the “achieveGoal” entry point. The notion of entry points and the corresponding types of execution semantics are explained in detail in [4].
3.2 Visualisation GUI

The Visualisation GUI enables the user to see the current status of a running WSMX server. The version currently available online receives status updates from the WSMX server at the UIBK, but for an interested target audience, the necessary files have been provided with the software packages for this deliverable. (for guidelines how to set up the visualisation, refer to Appendix 1)

The visualisation itself currently consists of a single web application, which shows the diverse phases of the execution semantics for the “achieveGoal” entry point, highlighting the currently active phase. To actually see the WSMX server change its status, a message has to be sent to the server, using the server’s management console (as was detailed in the previous section).
3.3 Additional documentation

The javadoc documentation for the WSMO API and its reference implementation wsmo4j can be found at http://wsmo4j.sourceforge.net/reference.html.

In addition the current documentation for the WSMX Integration API is included in the relevant download on the WSMX project at SourceForge, while the basis for the Integration API can be found in the DIP deliverable on the component APIs [3], respectively. A detailed description of the prototype architecture, the diverse entry points and the correlating execution semantics is described in [4].

The relevant javadocs have also been included in the software deliverable on CD.

4 CONCLUSION

This deliverable provides an overview of the first version of the DIP architecture prototype including detailed installation guidelines and documentation. An instance of the prototype can be accessed online at http://wsmx.deri.org:7000/. For demonstration purposes a local installation following the installation guidelines described in this deliverable might be useful. For demonstrations we recommend also the visualisation GUI. The guidelines for the local installation of the visualisation GUI are provided in the appendix.

A possible use of the current version of the prototype, as well as the necessary backend services, and front-end GUI, can be seen in [6].
REFERENCES


APPENDIX 1: WSMX VISUALISATION

In order to set up the optional WSMX visualisation for a local installation of WSMX, a number of additional products are needed. You need an application server or servlet engine, such as Tomcat, to deploy the web service. The web service has been tested on Tomcat 5.5.9 and Axis 1.2.1, but should work with any servlet engine which implements the version 2.2 or greater of the servlet API. Below is a sample list of possible products used to deploy the visualisation web service.

- Tomcat servlet engine 5.5.9 (Download from http://jakarta.apache.org/tomcat/index.html)
- Axis 1.2.1 (Download from http://ws.apache.org/axis/index.html)
- Java Mail API 1.3.2 (Download from http://java.sun.com/products/javamail/)

Follow the installation guidelines for Axis to set up the necessary software and deploy the web service, which can be found in the package wsmx_visualisation.zip. In addition to the web service, the visualisation web application (consisting of a servlet, a jsp page, its stylesheet and images) is included in the visualisation package. These have to be deployed to the Tomcat server as well.

Finally a small database has to be set up, to persistently store the status messages. The relevant database connection parameters have to be provided by configuring the “visualisation.properties” settings file in the web application package.

Detailed step-by-step installation guidelines can be found in the Readme file supplied with the visualisation package.
APPENDIX 2: THIRD PARTY SOFTWARE

Table 4 gives an overview of the third party software and libraries included with this release of WSMX, together with their licence information. All of these components are provided in the wsmx-thirdparty-0.2 package as well as in the wsmx_all-0.2-src package of the WSMX 0.2 release.

<table>
<thead>
<tr>
<th>Third Party Software (version)</th>
<th>Licence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant (1.5.2 and 1.6.2)</td>
<td>Apache Licence, Version 2.0; <a href="http://www.apache.org/licenses/">http://www.apache.org/licenses/</a></td>
</tr>
<tr>
<td>Axis (1.2_RC2)</td>
<td>Apache Licence, Version 2.0; <a href="http://www.apache.org/licenses/">http://www.apache.org/licenses/</a></td>
</tr>
<tr>
<td>Castor (0.9.5.4)</td>
<td>Copyright 1999-2003 (C) Intalio Inc.</td>
</tr>
<tr>
<td>cglib (2.0.2)</td>
<td>Apache Licence, Version 2.0; <a href="http://www.apache.org/licenses/">http://www.apache.org/licenses/</a></td>
</tr>
<tr>
<td>DOM4J (1.5.2)</td>
<td>Copyright 2001-2004 (C) MetaStuff, Ltd.</td>
</tr>
<tr>
<td>Hibernate (2.1.8 and 3.0)</td>
<td>GNU LESSER GENERAL PUBLIC LICENSE, Version 2.1</td>
</tr>
<tr>
<td>Hsqldb (1.7.3.3)</td>
<td>Copyright (c) 1995-2000 by the Hypersonic SQL Group</td>
</tr>
<tr>
<td>Hypercup (0.1)</td>
<td>J2SSH is dual licensed; distributed under an Apache style license or the LGPL</td>
</tr>
<tr>
<td>J2SSH</td>
<td>GNU LESSER GENERAL PUBLIC LICENSE, Version 2.1</td>
</tr>
<tr>
<td>Jboss 4.0 (Buildmagic tasks module)</td>
<td>GNU LESSER GENERAL PUBLIC LICENSE, Version 2.1</td>
</tr>
<tr>
<td>jdbc-stdext (2.0)</td>
<td>Sun Microsystems, Inc. Binary Code License Agreement</td>
</tr>
<tr>
<td>JINI (2.0.2)</td>
<td>Apache Licence, Version 2.0; <a href="http://www.apache.org/licenses/">http://www.apache.org/licenses/</a></td>
</tr>
<tr>
<td>JTA (1.0.1b)</td>
<td>Sun Microsystems, Inc. Binary Code License Agreement</td>
</tr>
<tr>
<td>Kaon 2.0: wsml4kaon2-0.1</td>
<td>The Apache Software License, Version 1.1</td>
</tr>
<tr>
<td>log4j (1.2.8)</td>
<td>Apache Licence, Version 2.0; <a href="http://www.apache.org/licenses/">http://www.apache.org/licenses/</a></td>
</tr>
<tr>
<td>odmg (1.0.3)</td>
<td>Apache Licence, Version 2.0; <a href="http://www.apache.org/licenses/">http://www.apache.org/licenses/</a></td>
</tr>
<tr>
<td>Software</td>
<td>License</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Spring (1.1.2)</td>
<td>Apache Licence, Version 2.0;</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.apache.org/licenses/">http://www.apache.org/licenses/</a></td>
</tr>
<tr>
<td>Catalina-ant (5.5.8)</td>
<td>Apache Licence, Version 2.0;</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.apache.org/licenses/">http://www.apache.org/licenses/</a></td>
</tr>
<tr>
<td>Wordnet (2.0)</td>
<td>Copyright 2003 by Princeton University</td>
</tr>
<tr>
<td>wsdI4j (1.5)</td>
<td>Common Public License - v 1.0</td>
</tr>
<tr>
<td>wsf (2.0)</td>
<td>Apache Licence, Version 2.0;</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.apache.org/licenses/">http://www.apache.org/licenses/</a></td>
</tr>
<tr>
<td>wsmo4j (0.2.0, 0.3.1 and 0.3.2)</td>
<td>GNU LESSER GENERAL PUBLIC LICENSE, Version 2.1</td>
</tr>
<tr>
<td>wsmoapi (0.2.0.1)</td>
<td>GNU LESSER GENERAL PUBLIC LICENSE, Version 2.1</td>
</tr>
<tr>
<td>Xdoclet (2.1.1)</td>
<td>Copyright (c) 2000-2004, XDoclet Team</td>
</tr>
<tr>
<td>Xerces (2.6.2)</td>
<td>The Apache Software License, Version 1.1</td>
</tr>
<tr>
<td>XSB (2.7.1), Flora (2-0.74)</td>
<td>GNU LIBRARY GENERAL PUBLIC LICENSE</td>
</tr>
</tbody>
</table>