



WSRF2OWLS DEVELOPER MANUAL

WP4

Document Filename:	KWF-WP4-IISAS-v1.0-WSRF2OWLSUserManual.doc
Work package:	WP4
Partner(s):	CYFRONET, IISAS
Lead Partner:	IISAS
Document classification:	CONFIDENTIAL

Abstract: This documentation presents a framework, which can semi-automatically generate the OWL-S descriptions for both stateful and stateless services based on the Web Service Description Language (WSDL) and corresponding annotations. Such functionality is inevitable in the grid environment hosting a vast number of services, which have to be semantically described in order to enable automated discovery, composition and invocation.



Delivery Slip

	Name	Partner	Date	Signature
From	Marian Babik	IISAS	30/12/2005	
Verified by	Piotr Nowakowski	CYFRONET	07/01/2006	
Approved by	Steffen Unger	FIRST	12/01/2006	

Document Log

Version	Date	Summary of changes	Author
0.1	10/11/2005	First version	Marian Babik
1.0	30/12/2005	Reviewed version	Marian Babik

CONTENTS

1. COPYRIGHT NOTICE	4
2. INTRODUCTION.....	5
2.1. ABBREVIATIONS AND ACRONYMS	5
3. IMPLEMENTATION STRUCTURE	6
3.1. PROTOTYPE USE CASES	6
3.2. PROTOTYPE COMPONENT MODEL.....	7
3.3. DETAILED IMPLEMENTATION MODEL.....	8
3.3.1. <i>Class WSRF2OWLS</i>	9
3.3.2. <i>Class WSRFTranslator</i>	10
3.3.3. <i>Class WSRFResourceProperties</i>	12
4. PROTOTYPE TESTING	14
5. CONTACT INFORMATION AND CREDITS.....	15
6. THE EDG LICENSE AGREEMENT	16

1. COPYRIGHT NOTICE

Copyright (c) 2005 by IISAS and K-Wf Grid. All rights reserved.

Use of this product is subject to the terms and licenses stated in the EDG license agreement. Please refer to Section 6 for details.

This research is partly funded by the European Commission IST-2002-511385 Project “K-WfGrid”.

2. INTRODUCTION

Web Service Resource Framework (WSRF) is a recent effort of the grid community to facilitate modelling of the stateful services. Design and development of the WSRF service based systems is quite common and there are several emerging WS initiatives, which try to automate the process of discovery, composition and invocation of such services. The semantic web services are a typical example, showing the potential of how ontological modelling can improve the shortcomings of the service oriented computing. One of the major obstacles in the process is the development of the ontologies, which describe web and grid services. Although, there are numerous standards for modelling semantic services, there are very few frameworks and tools, which can help automate the process of generating the semantic descriptions of services.

This documentation presents a framework, which can semi-automatically generate the OWL-S descriptions for both stateful and stateless services based on the Web Service Description Language (WSDL) and corresponding annotations. Such functionality is inevitable in the grid environment hosting a vast number of services, which have to be semantically described in order to enable automated discovery, composition and invocation.

2.1. ABBREVIATIONS AND ACRONYMS

WSRF Web Service Resource Framework

WSDL Web Service Description Language

WS Web Service

OWL Web Ontology Language

OWL-S Upper Ontology for Web Services

3. IMPLEMENTATION STRUCTURE

3.1. PROTOTYPE USE CASES

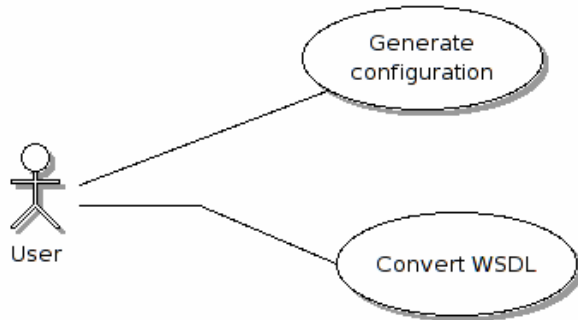


Figure 1 Use cases for WSRF2OWL-S

The following Use Cases were identified for the WSRF2OWL-S tool (as shown in Fig. 1):

- Generate configuration – allows the user to generate initial configuration for the given WSDL. The user has to add the corresponding concept mappings to the configuration manually.
- Convert WSDL – allows the user to convert the given WSDL to OWL-S according to the provided configuration

3.2. PROTOTYPE COMPONENT MODEL

The main components of the architecture are WSRF2OWL-S engine, WSRF2OWL-S translator and GOMOWL-S API. The overall architecture is depicted in Fig. 2.

The translation procedure is quite complex due to the complexity of the WSDL and OWL-S standards. It starts with a configuration file and an URL of the WSDL document. The translator parses the corresponding WSDL document extracting the operations, port-types, inputs, outputs and XML Schema types. A combination of the WSDL4J, Axis WSDL and GT4 WSDL utilities are used in the process. The translator then generates for each WSDL operation a skeleton of the OWL-S document. Furthermore, it creates the OWL-S inputs, outputs, preconditions and effects and maps the elements to the ontological concepts defined in the configuration file. If needed, it will create ontology with the domain ontological concepts using Jena API. The outcomes of the process are OWL-S documents describing the web service operations.

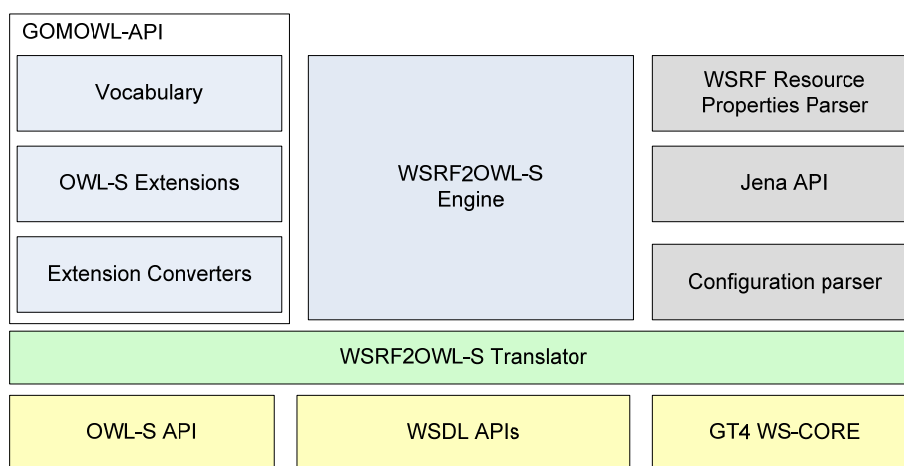


Figure 2 Components of the WSRF2OWL-S

The central component of the architecture is the WSRF2OWL-S engine, which configures and coordinates the translation procedure. The translator is used together with WSDL and OWL-S APIs to parse the WSDL document and generate the basic OWL-S structures. These basic structures are then put together by the engine into the consistent OWL-S document. The translator also allows adding extensions to it by using the GOMOWL-S API. The purpose of this API is to enable the user to create his own OWL-S language structures, such as domain specific OWL-S Profile, and seamlessly integrate the extensions into the translation process.

The translation of the WSDL description of a sample service MM5 is shown in Fig. 3. Since each service has multiple operations, the semantic descriptions are generated for each operation, thus enabling the possibility to create workflows of service operations. In the example a sample operation configureFrom-Properties is shown. Apart from WSDL description the translation process also needs configuration, which describes the mapping of the WSDL inputs/outputs to the domain ontological concepts.

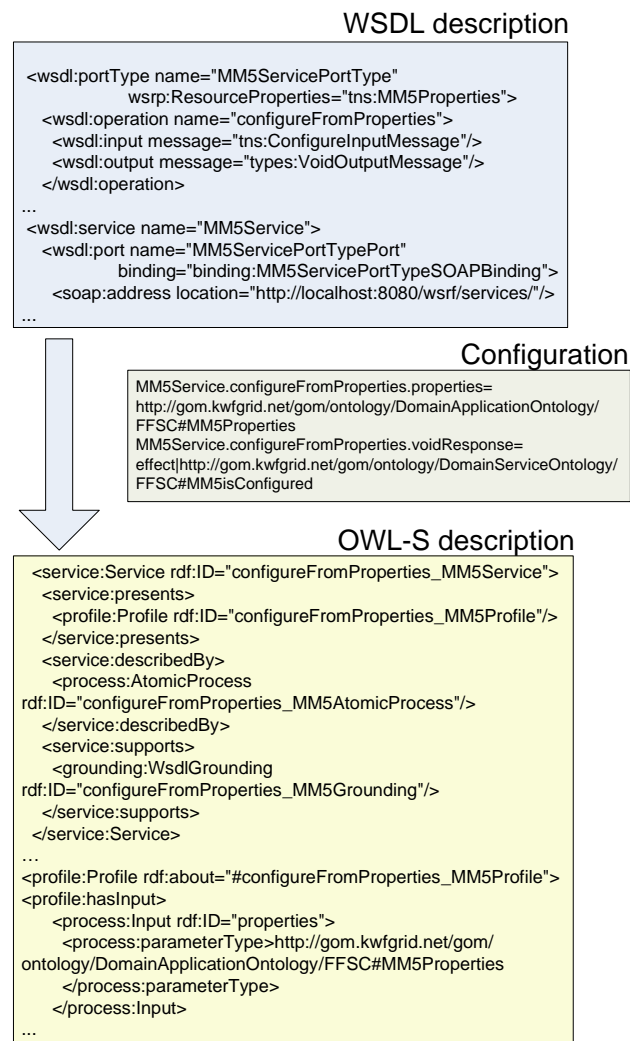


Figure 3 Sample translation of the MM5 configureFromProperties operation

Apart from flood forecasting simulations, WSRF2OWL-S was successfully used in generating the semantic description of services for the enterprise resource planning and coordinated traffic management applications.

3.3. DETAILED IMPLEMENTATION MODEL

The implementation model has two main packages, the first one is the WSRF2OWL-S core (engine and translator) and the second one is the GOMOWL-API, which is used to implement extension to the OWL-S API. WSRF2OWL-S core has three main classes:

- WSRF2OWL-S – provides the command line interface, parses the WSDL document and calls the WSRFTranslator, which creates the OWL-S documents
- WSRFTranslator – is based on the OWL-S API's WSDLTranslator and provides all the necessary methods for creating the OWL-S model and all its elements including OWL-S Profile, Process and Grounding
- WSRFResourceProperties – is a class, which implements special parsing methods needed to extract the WSRF Resource Properties from the WSDL documents

The overall summary for these three main classes is show in Fig.4 and described in detail in the following sections.

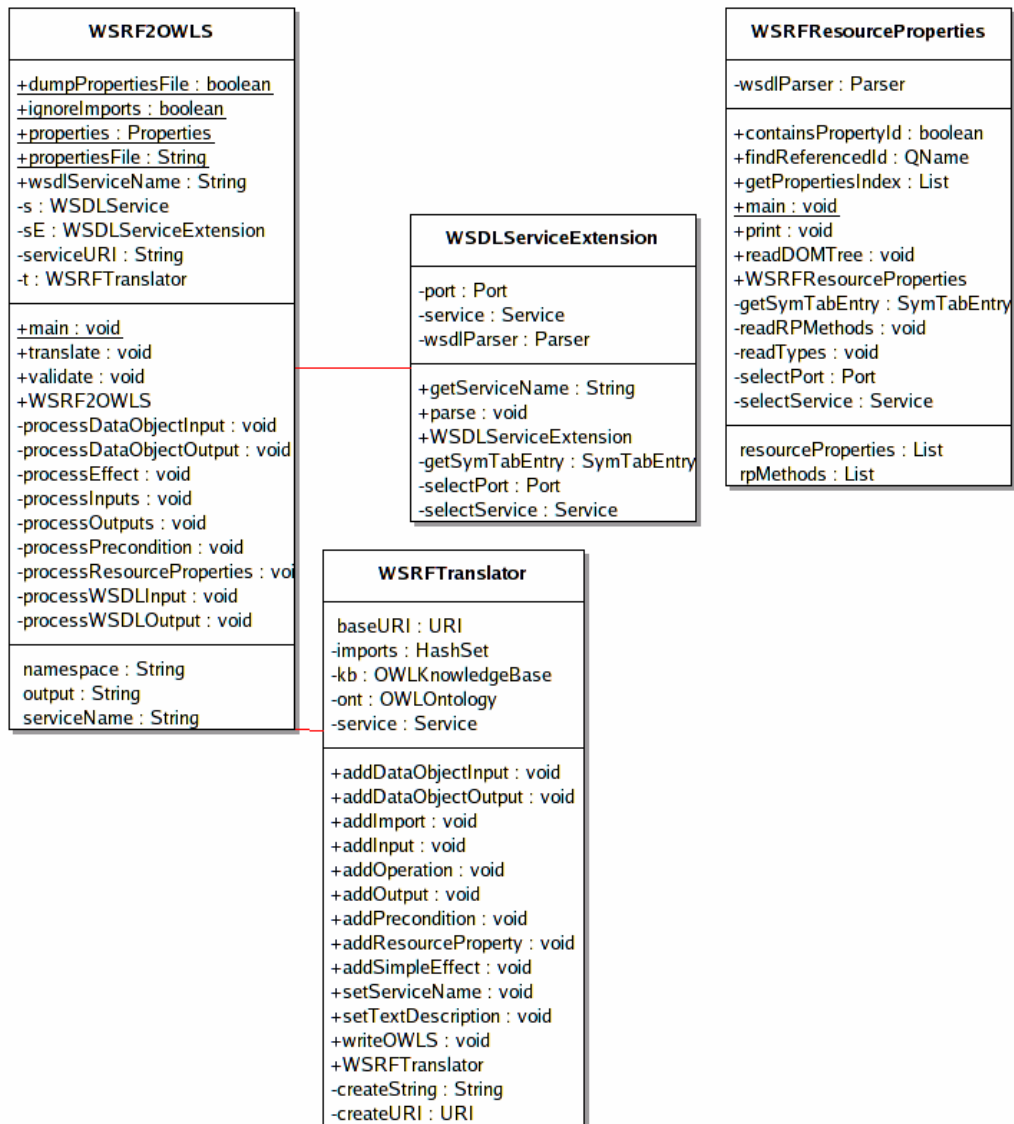


Figure 4 UML diagram of the basic classes

3.3.1. Class WSRF2OWLS

```

java.lang.Object
|
+--net.kwfgrid.wsrf2owls.WSRF2OWLS
  
```

```

public class WSRF2OWLS
extends java.lang.Object
  
```

WSRF2OWL-S main class The aim of the class is to provide command line interface, execute the parsing of the WSDL and call the corresponding methods of the WSRFTranslator

Constructor Summary

[WSRF2OWLS](#)(java.lang.String uri)

Default constructor expects a URL of the WSDL file, which should be processed

Method Summary

static void	main (java.lang.String[] args) WSRF2OWL-S tool main method Usage: WSRF2OWLS [-i input_uri e.g.
void	setNamespace (java.lang.String namespace) This method sets the OWL namespace for the final OWL-S document
void	setOutput (java.lang.String output) Sets the output directory for the generated documents
void	setServiceName (java.lang.String serviceName) This method sets the service name according to the WSDL specification; it can be also specified in the command line arguments
void	translate () Main method of the class, responsible for parsing and calling the process methods for each identified configuration element.
void	validate () Validates the generated OWL-S document

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

3.3.2. Class WSRFTranslator

```
java.lang.Object
|
+--net.kwfgrid.wsrf2owls.WSRFTranslator
```

```
public class WSRFTranslator
```

```
extends java.lang.Object
```

WSRFTranslator generates the OWL-s document

Constructor Summary

[WSRFTranslator](#)(java.lang.String serviceURI, java.lang.String prefix)
Constructor will create the default model and adds all the necessary converters.

Method Summary

void	addDataObjectInput (org.mindswap.wsd1.WSDLParameter param, java.lang.String paramName, java.lang.String paramConfig) Creates OWL DataObjectInput
void	addDataObjectOutput (org.mindswap.wsd1.WSDLParameter param, java.lang.String paramName, java.lang.String paramConfig) Creates OWL DataObjectOutput
void	addImport (java.net.URI url) Adds import statement to the OWL-S document.
void	addInput (org.mindswap.wsd1.WSDLParameter param, java.lang.String paramName, java.net.URI paramType, java.lang.String xsltTransformation) Creates simple OWL-S Input
void	addOperation (org.mindswap.wsd1.WSDLOperation op, java.lang.String prefix) Creates atomic process, grounding for the WSDL operation
void	addOutput (org.mindswap.wsd1.WSDLParameter param, java.lang.String paramName, java.net.URI paramType, java.lang.String xsltTransformation) Creates simple OWL-S Output
void	addPrecondition (org.mindswap.wsd1.WSDLParameter param, java.lang.String paramName, java.net.URI paramType) Create OWL-S precondition.
void	addResourceProperty (RP rpInstance)
void	addSimpleEffect (org.mindswap.wsd1.WSDLParameter param, java.lang.String paramName, java.net.URI paramType) Creates OWL-S precondition with simple effect
void	setServiceName (java.lang.String serviceName) Sets the service name, which is used to generate the OWL-S Profile rdf:ID
void	setTextDescription (java.lang.String textDescription) Sets the text description for the service.
void	writeOWLS (java.io.Writer out) Writes the OWL-S document to file

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

3.3.3. Class WSRFResourceProperties

```
java.lang.Object
|
+--net.kwfgrid.wsrf2owls.WSRFResourceProperties
```

```
public class WSRFResourceProperties
extends java.lang.Object
```

Constructor Summary

[WSRFResourceProperties](#)(java.lang.String wsdlURL)

Method Summary

boolean	containsPropertyId (java.util.Collection elementCollection, javax.xml.namespace.QName propertyId)
javax.xml.namespace.QName	findReferencedId (java.util.Collection elementCollection, javax.xml.namespace.QName propertyId)
java.util.List	getPropertiesIndex ()
java.util.List	getResourceProperties ()
java.util.List	getRpMethods ()
static void	main (java.lang.String[] args)
void	print (org.w3c.dom.Node node, java.io.PrintStream out)
void	readDOMTree (org.w3c.dom.Node n)

Methods inherited from class java.lang.Object
--

<code>equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait</code>
--

4. PROTOTYPE TESTING

All of components of the tool package are being subject to unit test's developed in parallel with the tool code base, and run using JUnit. The WSRF2OWL-S package components have been tested for their functionality during the development of the semantic descriptions of services for the:

- flood-forecasting simulations
- coordinated traffic management application
- enterprise resource planning

5. CONTACT INFORMATION AND CREDITS

Marian.Babik@saske.sk

Bkryza@icsr.agh.edu.pl

6. THE EDG LICENSE AGREEMENT

Copyright (c) 2004 K-WfGrid. All rights reserved.

This software includes voluntary contributions made to K-WfGrid. For more information on K-WfGrid, please see <http://www.kwfgrid.net>.

Installation, use, reproduction, display, modification and redistribution of this software, with or without modification, in source and binary forms, are permitted. Any exercise of rights under this license by you or your sub-licensees is subject to the following conditions:

1. Redistributions of this software, with or without modification, must reproduce the above copyright notice and the above license statement as well as this list of conditions, in the software, the user documentation and any other materials provided with the software.
2. The user documentation, if any, included with a redistribution, must include the following notice: "This product includes software developed by K-WfGrid (www.kwfgrid.net).” Alternatively, if that is where third-party acknowledgments normally appear, this acknowledgment must be reproduced in the software itself.
3. The names "K-WfGrid" and "Knowledge Workflow Grid" may not be used to endorse or promote software, or products derived therefrom, except with prior written permission by steffen.unger@first.fraunhofer.de.
4. You are under no obligation to provide anyone with any bug fixes, patches, upgrades or other modifications, enhancements or derivatives of the features, functionality or performance of this software that you may develop. However, if you publish or distribute your modifications, enhancements or derivative works without contemporaneously requiring users to enter into a separate written license agreement, then you are deemed to have granted participants in K-WfGrid a worldwide, non-exclusive, royalty-free, perpetual license to install, use, reproduce, display, modify, redistribute and sub-license your modifications, enhancements or derivative works, whether in binary or source code form, under the license conditions stated in this list of conditions.

5. DISCLAIMER

THIS SOFTWARE IS PROVIDED BY K-WfGrid AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, OF SATISFACTORY QUALITY, AND FITNESS FOR A PARTICULAR PURPOSE OR USE ARE DISCLAIMED. K-WfGrid AND CONTRIBUTORS MAKE NO REPRESENTATION THAT THE SOFTWARE, MODIFICATIONS, ENHANCEMENTS OR DERIVATIVE WORKS THEREOF, WILL NOT INFRINGE ANY PATENT, COPYRIGHT, TRADE SECRET OR OTHER PROPRIETARY RIGHT.

6. LIMITATION OF LIABILITY

K-WfGrid AND CONTRIBUTORS SHALL HAVE NO LIABILITY TO LICENSEE OR OTHER PERSONS FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, OR PUNITIVE DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOSS OF USE, DATA OR PROFITS, OR BUSINESS INTERRUPTION, HOWEVER CAUSED AND ON ANY THEORY OF CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR OTHERWISE, ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.