(Draft) Lecture Notes in Computer Science (LNCS). 2008, vol. 5232, p. 375-384. ISSN 0302-9743.

A Method to Derivate SOAP Interfaces and WSDL Metadata from the OGC Web Processing Service Mandatory Interfaces

Gonzalo Sancho-Jiménez, Rubén Béjar, M.A. Latre, Pedro R. Muro-Medrano

Computer Science and Systems Engineering Department, University of Zaragoza,
María de Luna 1, E-50018 Zaragoza, Spain
{gsancho, rbejar, latre, prmuro}@unizar.es

Abstract. Web Processing Services (WPS) expose processing functionality using Web Service technology. The WPS specification describes the interfaces to publish geospatial processes on the Web. It includes a platform-neutral and several platform-specific versions of its interfaces. Some of the platform-specific interfaces are mandatory and others are optional. In this paper, we present a method to support the automatic derivation of the optional SOAP interfaces and WSDL metadata from the mandatory ones in any WPS. These interfaces can then be used to facilitate the chaining of the WPS with other Web Services, using for example BPEL, and to improve the interoperability of these services. In addition to that, we have created a tool to validate the proposed method.

1 Introduction

The Open Geospatial Consortium (OGC) Web Processing Services (WPS) specification [1] describes the interfaces needed to design a service in which geospatial process is offered on the Web. It includes a platform-neutral and several platform-specific versions of its interfaces. Some of the platform-specific interfaces are described as mandatory while some others are optional, nevertheless offering support to all of them is recommended. At the same time, multiple encoding ways for requests and responses are described, each of them appropriate for one or more platforms.

The aim of this paper is to propose a method to support the automatic derivation of the optional Simple Object Access Protocol (SOAP) interfaces and Web Services Description Language (WSDL) metadata from the mandatory ones in any WPS, as well as a tool to test its viability. These interfaces can then be used to facilitate the chaining of the WPS with other Web Services, using for example Business Process Execution Language (BPEL).

This method helps developers to avoid the tedious task of programming the different platform-specific interfaces to the geoprocessing operations encapsulated by the WPS. They only need to create the mandatory ones and the others are automatically generated. It also helps to avoid mistakes and unwanted differences among the different platform-specific interfaces to the same service. Then, it will be easily maintained and adapted to specific requirements.

The method can be applied to any compliant OGC WPS because they must support at least the mandatory parts of the specification. It does not require access to the source code or the binaries of the WPS, it only needs network access to the mandatory interfaces of the WPS.

The paper shows how the mandatory platform-specific interfaces define a model of the service, that can be used to automatically create actual instances of the service that support different technologies. The mandatory interfaces could be seen as the "canonical" specification which is used to generate its many optional interfaces, facilitating thus interoperability of the WPS. This would prevent problems coming from different behaviours of the service under its different interfaces.

The rest of the paper is organized as follows: section 2 summarizes related work including a WPS standard description. Section 3 describes the automatic derivation method to offer SOAP/WSDL support in a WPS. Then, section 4 tests the feasibility of the method showing an implemented tool and giving an example of use of this tool. The last sections introduce some ideas on future work and conclude.

2 Related work

There is a wide set of published works dealing with the issue of SOAP/WSDL adaptation method for OGC services. Those works use automatic adaptation methods which must then be checked.

In [2] there is an analysis of the use of BPEL in comparison to Web Services Modelling Ontology (WSMO) and a state-of-the-art composition approach. It states that current implementations of OGC services are some kind of hybrid Representational State Transfer (REST)-based services, whereas BPEL requires SOAP services, so OGC services, which do not provide SOAP interfaces, need a wrapper, which acts as a proxy to the OGC services. In this paper we review this wrapping method in order to support the automatic derivation from the optional interfaces to the mandatory ones.

The suitability of the Web Service Orchestration (WSO) technology as a possible solution for disaster management scenarios has been evaluated in [3]. It analyses and describes data format adaptation for OGC services with a proxy server acting as binary transcoding service. The proxy launches an OGC compliant request to the OGC service and receives the response from the service. The proxy stores the physical part of the response on a simple http-server directory and returns the Uniform Resource Identifier (URI) of the stored data to the BPEL process. Using a proxy method is a suitable idea for automatic adaptation. E The first steps in developing a Web service framework for heterogeneous environmental information systems are presented in [4]. With a framework which allows an easy assembly of data and method services which are distributed over the entire Web. The framework [4] uses the Web Service Framework, which allows users to create their own services by combining the existing ones. This frame-

(Draft) Lecture Notes in Computer Science (LNCS). 2008, vol. 5232, p. 375-384. ISSN 0302-9743.

work uses a generic adapter which allows a technical encapsulation of different middleware technologies.

The aim of [5] is to present a discussion of technology issues considered in an initiative of the OGC Interoperability Program. It is focused on creating general recommendations and guidelines for WSDL/SOAP support to existing OGC Web Services. In this paper these recommendations are followed as much as possible.

2.1 The WPS standard

The specified WPS provides client access to pre-programmed calculations or computation models that operate on spatially referenced data. The data required by the service can be delivered across a network, or be available at the server. The info can be provideded as image data or through data exchange standards such as Geography Markup Language (GML). The calculation can be as simple as subtracting one set of spatially referenced numbers from another (e.g. determining the difference in influenza cases between two different seasons), or as complicated as a global climate change model [1].

The WPS interface specifies three operations that can be requested by a client and performed by a WPS server and that are considered as mandatory and must be implemented by all WPS servers. These operations are: GetCapabilities, which allows a client to receive service information, DescribeProcess, which should return detailed information about server's processes and Execute, which runs a specified process implemented by the WPS. The mandatory platform-specific interface of these operations must be implemented and the optional platform-specific interface is recommended to be supported.

In table 1, each operation includes its optional and mandatory platform-specific interfaces. A compliant WPS must contain the mandatory platform-specific interface, although the implementation of the optional ones would improve interoperability with different clients.

Table 1. Optional and mandatory platform-specific interfaces

Operation	GET/KVP	POST/XML	SOAP
GetCapabilities	Mandatory	Optional	Optional
DescribeProcess	Mandatory	Optional	Optional
Execute	Optional	Mandatory	Optional

The WPS standard recommends including WSDL metadata for each operation. Despite being less comprehensive mechanism, WSDL makes dynamic binding to dynamic services easier. It offers a widely adopted alternative to the publishing mechanism built into the WPS interface specification.

The aim of this paper is to describe a method in which only the mandatory platform-specific interface has to be defined, while the others may be automat-

ically derived, allowing to add optional platform-specific interfaces to a WPS without requiring access to its source code or binaries.

This method allows all WPS to comply with World Wide Web Consortium (W3C) standards using SOAP/WSDL without code modification [5]. The reason for this mapping is the fact that certain developments only focus on supporting SOAP services and do not define GET and POST bindings. The reason for this is that this work heavily relies on mainstream standards such as Security Assertion Markup Language (SAML), eXtensible Access Control Markup Language (XACML) and the WS-* family which is oriented to support SOAP.

3 Automatic derivation method

In order to offer SOAP/WSDL support, we propose a method which combines several standardization efforts with a flexible and extensible Web service framework, supporting a higher interoperability between different resources.

It is necessary to create an intermediate proxy which adds a new SOAP/WSDL interface to the WPS that does not offer it. This proxy could remain separated or embedded in the processing service, being able to connect with the WPS through the mandatory interfaces. The proxy design consists of two modules. The first one generates the WSDL metadata through WPS request obtained data, and the second one adapts a SOAP request (generated from a WSDL metadata) to the mandatory WPS request.

Carrying out the adaptation task through a proxy allows us to incorporate it to the implementation of a WPS, no matter if it is under our control or not. If it is decided to use the adaptation proxy as part of a WPS which belongs to us, we can avoid the tedious chore of having to program the different interfaces over each one of the offered operations: it is only necessary to program the mandatory operations, which are the ones that provide more information.

Figure 1 shows an example of a proxy integration into a WPS in order to define a new SOAP interface. The mentioned WPS contains only a definition of the offered services and only an interface per public-accessible operation. The proxy accesses internally (through the mandatory interface) to the WPS operations offering a new WSDL/SOAP interface.

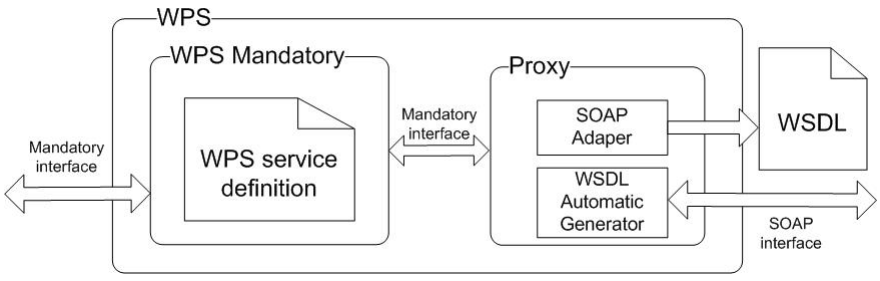

Fig. 1. Example of proxy integration

(Draft) Lecture Notes in Computer Science (LNCS). 2008, vol. 5232, p. 375-384. ISSN 0302-9743.

3.1 WSDL automatic generation

The WSDL automatic generation method provides the WSDL metadata document by using data obtained WPS requests, following through the WPS structure as far as possible. The WSDL generated describe the WPS SOAP interface to provide it over the Internet. A client program connecting to a WPS can parse the WSDL to determine what functions are available on the server.

The WSDL generation request for a WPS is sent directly to the proxy URL which does not know the WPS URL. That is the reason why the WPS URL needs to be included as a parameter of the request. With this URL the proxy may perform the necessary requests against the WPS using the mandatory interface to compose the WSDL.

Following, as far as possible, the best practices of the OGC WPS 1.0.0 specification, the WSDL service description can be requested as a single WSDL document (including all the operations) or as a document per operation. An example of a WSDL request for all the WPS operations as a single WSDL document is shown in figure 2.

```
http://URLProxy?URLWPS=http://URLwps&WSDL
```

Fig. 2. WSDL request for all the WPS operations

Attending to the WSDL request of an individual process, it keeps the same structure including the operation identifier as show in figure 3.

```
http://URLProxy/identifier?URLWPS=http://URLwps&WSDL
```

Fig. 3. WSDL request for individual processes

Each WSDL request contains a GetCapabilities and a DescribeProcess methods specification, but as many Execute method specifications (with their own parameters) as operations offered in the WPS. Figure 4, shows that GetCapabilities and DescibeProcess requests contain the parameters described in the WPS specification but are codified as SOAP parameters. GetCapabilities operation includes (in WSDL document) the parameters AcceptVersions and language and in case of the DescribeProcess operation, version, language and identifier are included.

The responses received from these two operations comply with the XML Schema Definition (XSD) defined in the standard but with the difference that these responses are wrapped as a SOAP response.

In order to fit the WPS request optimally and in an automatic way, each of the parameters of the Execute operation have to be defined as much as possible in the WSDL document. The DescribeProcess response is parsed to obtain the

```
<xsd:element name="GetCapabilities">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element minOccurs="0" name="acceptedVersion"
     type="xsd:string"/>
      <xsd:element minOccurs="0" name="language" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:elememnt name="DescribeProcess">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element minOccurs="1" name="identifier"
     maxOccurs="unbounded" type="xsd:string"/>
      <xsd:element minOccurs="1" name="version" type="xsd:string"/>
      <xsd:element minOccurs="0" name="language" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Fig. 4. GetCapabilities and Describe method specification in the WSDL generated

necessary information required to generating the WSDL Execute request and response; each input and output is codified as a different element in the SOAP body using the identifier as name and the most similar type defined in the WSDL specification. The operations containing state=true are enabled, but they are added as if they do not have that option; this is due to the fact that, in the prototype, no mechanism allowing this request is implemented while the option of allowing the calculation is intended.

Each of the DescribeProcess response parameters are included into the response and are reconverted into a WSDL element, in which the number of occurrences is specified with a maximum and a minimum limitation. This element contains a complexType field composed with each one of the DescribeProcess response attributes specifying their possible and default values. Code example shows an automatic conversion from WPS ComplexType specification to WSDL specification.

As seen in figure 5, additional information is included specifying, for example, diverse parameters in the request attributes. In this example, just with the WSDL specification, it is possible to know that in the complexData attribute, it has to be specified an encoding attribute as well as a schema attribute (which has a default value); it can also be deduced that the output data are to be codified in a 64 base, and also, that the attribute may appear zero or more times in the request.

```
<complexType name="IDOperation_IDParameter_complex">
  <sequence>
    <element name="encoding" type="xsd:anyURI"  />
    <element name="schema" type="xsd:anyURI"
   default="http://schemas.opengis.net/wfs/1.1.0/wfs.xsd" />
    <element name="data" type="xsd:base64Binary" />
  </sequence>
</complexType>
<element name="IDOperation_IDParameter_type">
  <complexType>
    <sequence>
      <element name="input"
     type="tns:IDOperation_IDParameter_complex" minOccurs="0"
     maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
```

Fig. 5. Execute method specification in WSDL

3.2 Automatic SOAP adaptation

Once the WSDL has been generated, this can be used by any user as a description to generate requests to the SOAP WPS interface. The requests are sent to the proxy URL transparently to the user. The WPS URL is contained in the proxy URL like a parameter towards which the operation is finally going to be performed as seen in figure 6.

```
<wsdl:service name="OperacionesService">
  <wsdl:port name="\textbf{ServicioOperaciones}" binding="impl:WPS_Binding">
    <wsdlsoap:address
   location="http://URLProxy?URLWPS=http://URLwps"/>
  </wsdl:port>
</wsdl:service>
```

Fig. 6. URL WPS include in WSDL

The proxy parses the SOAP request to obtain its attributes, values and the WPS URL. Once it is done, the proxy generates the request following the mandatory interface and sends it to the WPS. For example by POST in case of Execute, and by Key Value Pairs (KVP)/GET in case of GetCapabilities and DescribeProcess. When the WPS has performed the required operations, the response is generated and sent to the proxy. This proxy parses the XML parameters and generates a SOAP response that corresponds to the one defined in the WSDL document.

The automatic adaptation SOAP service can be located in a different server to the WSDL generator due to fact that they are two independent services. SOAP adapter works in an independent form and without having information about the request WSDL description which is going to adapt, making a streaming adaptation. This automatic adaptation is possible thanks to the above guidelines on the generation of WSDL description types, due to the fact that these types contain enough information to generate a valid request to the WPS from the SOAP request.

4 Testing tool

Following the described method, a tool has been designed to test its viability. This tool provides the mentioned features and can be used to add new WPS platform-specific interfaces to any WPS.

As shown in figure 7, the proxy generates the WSDL metadata in accordance with the WPS and performing the request conversion between SOAP and the mandatory WPS platform-specific operation. The proxy is completely transparent for the WPS client, so it can use it as another service interface.

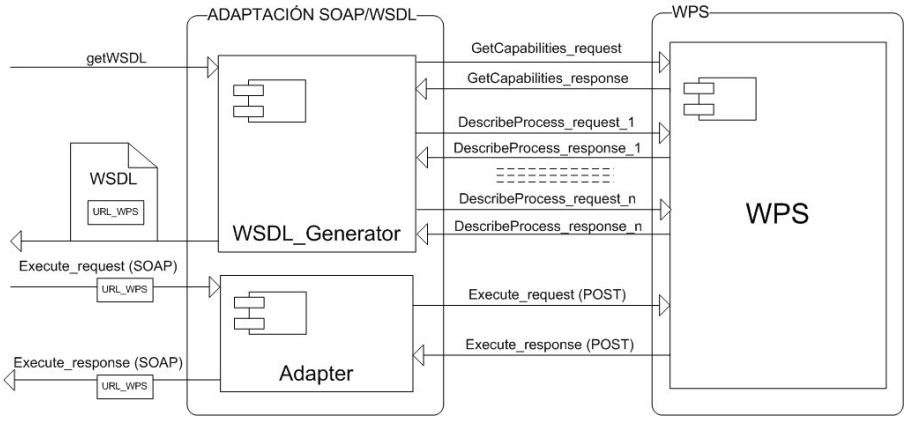

Fig. 7. Proxy design

The first steep is the WSDL generation: the proxy needs to know all the operations that the WPS offers in order to generate the WSDL description; with this aim a GetCapabilities request is made using the standard (GET/KVP). Once all the operations are known, a DescribeProcess request is performed (GET/KVP) per each operation to know all operation parameters. With these data and a collection of guidelines justified above, it is possible to build a WSDL in an automatic way; the second steep is SOAP adaptation: the tool simply follows the proposed method to SOAP encoding and decodes the requests and responses, providing that WSDL metadata has been well defined.

Both WSDL generator and SOAP adapter services have been implemented in two different Java servers without communication between them. They perform the SOAP and WSDL encoding and decoding using Java libraries. However, it is not the purpose of this paper to describe the low level implementation of this tool.

4.1 Composition example

To give an example of the use of this tool, a WPS without SOAP interface has been adapted in order to be chained with BPEL engine. The final target operation is an intersection between a vectorial and a raster from several operations of the same WPS. This series of operations can be chained in a simple and fast way, allowing creating more complex operations with scalability. For the BPEL composition, only Execute methods of each sub-operation are needed, since GetCapabilities and DescribeProcess methods list and describe the operations.

The Spanish Spatial Data Infrastructure (IDEE) WPS are used with the purpose of testing this tool. It has been implemented using the degree [6] framework which uses the 0.4.0 WPS version. This implementation does not provide the optional SOAP/WSDL interfaces. IDEE develops the operations vectRastConversion, which converts a vector data to a raster data, and rasterIntersect, which intersects two raster maps in a new one.

In order to make the BPEL's composition operation has been chosen the use of Apache Orchestration Director Engine (ODE) [7] which executes business processes written following the BPEL standard. It talks to Web services, sending and receiving messages, handling data manipulation and error recovery as described by the process definition. It supports both long and short living process executions to orchestrate all the services that are part of an application.

In figure 8 a scheme of the operation desired to make can be seen. In the first WPS operation the conversion from a vectorial map to a raster one is made, next the intersection with other raster is done. These operations can be independently reused as belonging to a complex processing service.

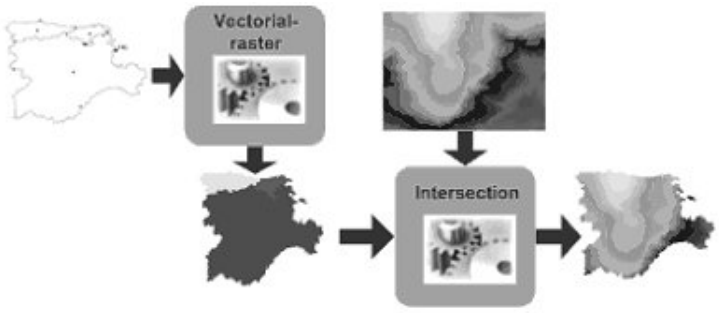

Fig. 8. Scheme of the operation

BPEL's process that composes the operations in figure 9 consists of a sequence that includes all the described operations since it is a not parallelizable operation. The results of each sub-operation are assigned to the inputs in the next operation with the necessary parameters that are defined in the process input. Each one of the sub-operations makes a SOAP request to an adapter in an automatic way and following the WSDL scheme that has been used to make the composition. In the adapter server, each SOAP request is processed in different instances. In each instance, the requests are decoded and the POST Execute request is composed to be sent to the specified WPS. When the WPS execution has been finished a POST response is sent to the Adapter which encodes the SOAP response and sends it to the BPEL process to continue the execution.

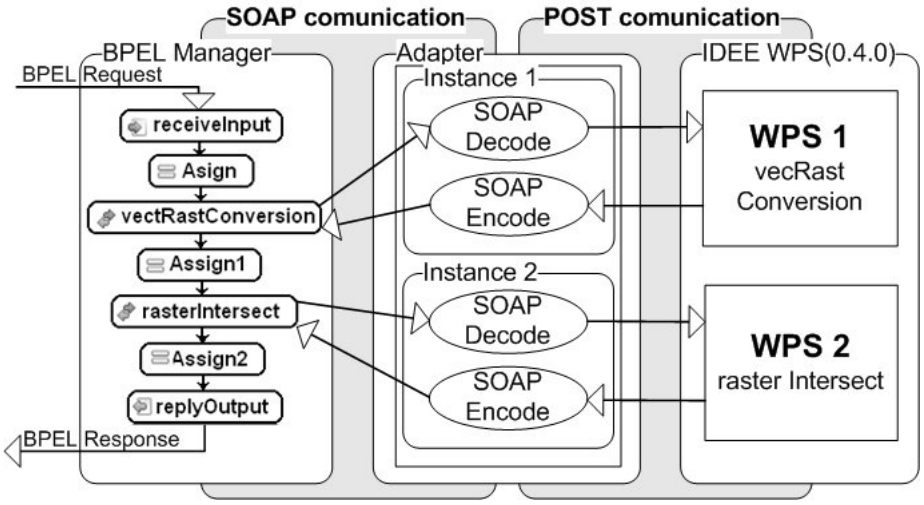

Fig. 9. BPEL's process

5 Conclusions

Processing services provide a new way of offering geoprocessing capabilities adapted to different user needs. Interaction among processing services is possible, in spite of the fact that they are independent, providing a new way to create new complex geoprocessing operations.

The WPS standard opens the possibility to implement several platform-specific interfaces as SOAP/WSDL needed by mainstream standards such as SAML, XACML and the WS-* family. The method presented in this paper allows adding the optional SOAP platform-specific interfaces to any WPS, without requiring access to the source code or the binaries of this service; it only needs network access to its mandatory interfaces. Using this method in order to provide optionals interfaces, it is possible to avoid errors that could be committed when

programming different platform-specific interfaces to offer the same functionality. It also makes it easier to integrate WPS with other Web Services, facilitating thus interoperability.

In spite of the fact that the described method follows as much as possible the OGC SOAP recommendations [5], new additional information has been added using all WSDL and SOAP implementation resources in order to approach SOAP and mandatory requests. For example, input parameter types, units of measure, default values, formats, schemes and encodings have been added instead of an undefined list of String parameters as recommended [5]. It allows automatic derivation but it also allow increasing the WSDL description information provided. To allow SOAP adaptation without information stored, the WPS URL has been added to the SOAP description, consequently to the SOAP request that has been added too.

A tool has been designed to validate the presented method. The designed tool works correctly and, to test it, a BPEL composition, using IDEE WPS, has been implemented due to the fact that IDEE WPS has not got any SOAP/WSDL interface. The incorporation of this tool in IDEE WPS is currently being studied.

6 Acknowledgements

The basic technology of this work has been partially supported by the Spanish Ministry of Science and Technology through the project TIN2007-65341 from the National Plan for Scientic Research, Development and Technology Innovation, by the University of Zaragoza through the project UZ200700024 and by GeoSpatiumLab S.L.

References

1. OGC: Standard Description of the CGDI, Web Processing Service (WPS). OGC publicly available tandard, Open Geospatial Consortum (2007)
2. Moses Gone, S.S.: Towards Semantic Composition of Geospatial Web Services Using WSMO in Comparison to BPEL. Interoperability and spatial processing in GI applications (2007) 43–63
3. A. Weiser, A.Z.: Web Service Orchestration (WSO) of OGC Web Services (OWS) for Disaster Management. 3rd International Symposium on Geoinformation for Disaster Management (Toronto, Canada) (2007) 33–41
4. Radetzki, U., Alda, S., Bode, T., Cremers, A.B.: First Steps in the Development of a Web Service Framework for Heterogeneous Environmental Information Systems
5. OGC: OWS 5 SOAP/WSDL Common Engineering Report version 1.0.0. Technical report, Open Geospatial Consortium, year = 2008, type = OGC Common Engineering Report,
6. Consortium, O.G.: Deegree. http://www.deegree.org/ (2008)
7. Apache: Orchestration Director Engine (ODE). http://ode.apache.org/ (2008)