Towards a Transactional Web Processing Service (WPS-T)

Bastian Schaeffer
Institute for Geoinformatics, University of Muenster
schaeffer@uni-muenster.de

Abstract. The OGC Web Processing Service specification provides a means to perform distributed web-based processing on geodata. However, the specification does not provide the ability to dynamically deploy and undeploy processes. This was the starting point for this paper to extend the specification with a generic means to deploy and undeploy processes at runtime, by adding two new operations to the WPS interface. Since the proposed approach allows any kind of processes to be deployed, specialized deployment profiles have to be offered. A BPEL deployment profile will be introduced, which allows geoprocessing workflows to be exposed as simple WPS processes. The applicability will be demonstrated by a real world air quality assessment use case.

1 Introduction

The rapid evolution from monolithic desktop GIS applications with tightly coupled geodata to Spatial Data Infrastructures (SDI) (Groot and McLaughlin 2000) with independent, interoperable and distributed Web Services has changed the GIS world fundamentally (Masser 2003) (Bernard et al. 2005). Nevertheless, existing SDIs are focused on data retrieval and visualisation (Kiehle et al. 2006). To generate information out of data, it becomes necessary to process data (Ernst 2000). With growing computational power and network capabilities, web based processing of distributed data towards information becomes therefore one of the main interests in the IT world. The advent of Web Services and Service-Oriented Architecture (SOA, (Gottschalk et al. 2002)) leveraged distributed processing using Web Service technology and became also one of the major interests in the geoinformation (GI) domain, as most of the GI applications involve large amounts of globally distributed data and as the demand for distributed available geo-information increases. The role of web-based geoinformation as a key factor for SDIs in the future requires sufficient concepts for web-based processing. Such processes have to be able to access globally distributed data and to provide the information in line with the already available standards. The OGC Web Processing Service (OGC 2007a) provides a standardized means for this purpose of

web-based geoprocessing. In the last two years a number of communities such as 52°North (Schaeffer and Foerster, 2007) or pyWPS (Cepicky 2006) have started to implement this specification and demonstrated its applicability by several use cases. While Foerster and Stoter (2006) focused on generalization, Kiehle (2006) proofed the WPS concept within the groundwater vulnerability measurement domain.

However, the complexity of geospatial analysis often requires multiple processing steps implemented by processing chains, in which each element performs an isolated task while the whole chain addresses a much broader problem (Gehlot and Verbree 2006). For the future, Alameh (2003) identified these service chains as one of the key concepts in enabling value-added chains in SDIs. Especially with the advent of the SOA paradigm, the modelling of SOA-based Geoproccessing workflows based on standardized OGC services will become more vital for the growing GIS community (Kiehle et al. 2006) (Weiser and Zipf 2007). Web Service Orchestration (Pelz 2003a) provides a means to represent these workflows based on XML. While van der Aalst (2003) and (Pelz 2003b) discuss different orchestration approaches and compare different languages for the IT world, the applicability of BPEL as an appropriate workflow language and thereby the applicability of the centralized orchestration approach for the GI-domain was demonstrated by (Weiser et al. 2006) and the OGC Open Web Service Testbeds 4 (Keens 2007).

Despite the importance of workflow modelling, only little research has been conducted on exposing or sharing process models in a standardized way. An innovative geoprocessing model is usually developed with a dedicated purpose but with limited testing. In most cases the workflow is published, but it is short lived since e.g. the researcher changes the employer, the projects ends or the coding platform is changed (Gehlot & Verbree 2006). This limits the interoperability since there is no standardized way for packaging geoprocessing workflows and exposing them in a standardized way for integration into SDIs. This was the starting point for this paper to create a standardized way for exposing SOA based geoprocessing workflows on the basis of a WPS. Unfortunately, the WPS interface does not offer an operation for this purpose. Therefore this paper proposes to extend the WPS interface with the ability to dynamically deploy and undeploy processes at runtime. The taken approach will not be limited to workflows only but will moreover present a generic way of dynamically deploying and undeploying processes. Therefore, different kinds of WPS-T profiles can be developed to foster interoperability. Although the ORCHESTRA project developed a similar service (ORCHESTRA 2007), the presented approach goes beyond the

ORCHESTRA service specification by allowing not only service chains to be deployed. Moreover it builds on top of the already standardized WPS interface and thus exposes the deployed processes as standardized WPS processes. This allows the reuse of existing clients for e.g. executing those processes. Additionally, by following a generic profiles approach, XML schemas can be used to validate the input data, while the ORCHESTRA Service only uses non standardized identifiers to point out the supported workflow languages.

However, the focus will be led on creating a specific profile for exposing complex models as simple WPS processes as the basic motivation for the taken approach. This would foster the integration of these models as building block for newly developed hierarchical models.

Section 2 provides a brief overview of the underlying technologies. This is followed by a conceptual design including the extended service specification and intended architecture. The last two chapters provides a proof of concept implementation in the field of air quality assessment.

2 Background

This chapter provides an overview of the underlying technology for this study. Especially the Web Processing Service and related geoprocessing workflows will be introduced.

2.1 OGC Web Processing Service

The proposed OGC Web Processing Service (WPS) Specification (OGC 2007a) describes a standardized way to perform distributed (geo) processes in a spatial data infrastructure. These processes can be as simple as the sum of two numbers (e.g. population) or as complex as a global climate model. The data required by the service can be delivered across a network or made available on a server. The specification is held generic in order to support any kind of data format. Thus, image data formats or data exchange standards such as Geography Markup Language (GML) (OGC 2007b) can be used for input or resulting data.

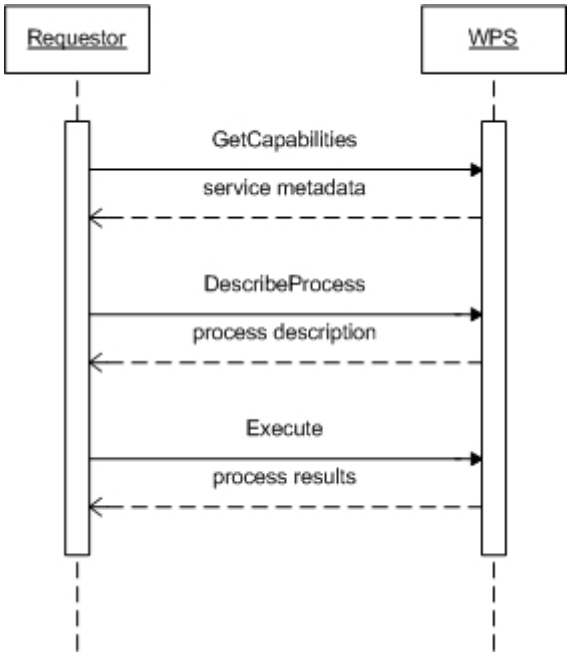

Figure 1: Basic WPS interaction pattern

The interface is based on three operations as shown in figure 1: *GetCapabilities*, *DescribeProcess* and *Execute*. Since a WPS should be able to be integrated in to the OpenGIS framework, it has to offer the GetCapabilites operation generally described by the OWS Common specification. Besides the OWS Common metadata, and basic service metadata, all processes offered by the WPS are briefly described in the capabilities document. The DescribeProcess operation offers full descriptions of a specific process. This includes the detailed input and output parameter descriptions as well as supported formats.

The WPS specification distinguishes between three basic input/output data types.

1. “ComplexData” such as XML (e.g. GML), imagery or a reference (URL) to the actual data.

2. “LiteralData”, with a specified “DataType”, allowed values, “DefaultValue” and “SupportedUOMs” indicated.

3. Bounding Box information, using one of the supported coordinate reference systems.

Using these input parameter values, requestors can perform a desired process offered by a server via the Execute operation. The WPS creates either a direct response to the request including information about the status of the process or, alternatively, the server can be directed to store the

result(s) as web accessible resources identified by a Uniform Resource Locator (URL). If the results are stored, the Execute response will consist of an XML document that includes a URL for each stored output, which the client can use to retrieve those outputs

2.2 Web based Geoprocessing Workflows

The chaining of web services to workflows where the output of a partner can be used as input for another partner has been identified as one of the key factors for SDIs (Alameh 2003). With advances in SOA and Web Services, it becomes possible to solve complex geoprocessings tasks by composing several web based processes offered by differed web service to a workflow. This workflow can be exposed again as a Web Service. Alonso et al. (2003) speaks of a composite service and names the act of combining Web Services, *Web Service Composition*. Service composition can be either performed by composing elementary or composite services, the latter being recursively defined as an aggregation (Khalaf et al. 2003) of elementary and composite services.

The services inside of a workflow can follow different interaction patterns: Web Service Orchestration (WSO) or Web Service Choreography. The first one is defined as a description of how composed Web Services interact on the message level (Pelz 2003a). This includes business-logic and execution order. The latter is more focused on the public message exchange between multiple parties (Reichert et al. 2004), while Web Service Orchestration has to follow the specified message exchange protocol, but adds the business logic as internal Web Service calls. Since Web Service Choreography specifies an explicit message interaction protocol, each service knows its predecessor and successor. In contrast, Web Service Orchestration is often realized by a central orchestration engine, which coordinates the interaction based on a predefined message exchange protocol. Hence, the Web Services can be held loosely coupled (Veerawarana et al. 2005).

The semi-automatic central orchestration architecture is implemented by several vendors (e.g. Oracle BPEL Process Manager[1] or ActiveBPEL[2]). These frameworks make use of the Business Process Execution Language (BPEL) as the de-facto standard (van der Aalst et al. 2005). BPEL evolved from former standards, such as graph based WSFL or block based and algebraic XLANG and was proposed by IBM, Microsoft and BEA

[1] Oracle BPEL Process Manager website: www.oracle.com/technology/bpel/index.html
[2] ActiveBPEL website: www.active-endpoints.com/active-bpel-engine-overview.htm

(Andrews et al. 2003). The language is XML based and describes the roles and partners involved in the message exchange, supported port types and orchestration information of a process. It can also be regarded as a service composition model (Wohed et al. 2003), which supports composition and coordination protocols (Chen et al. 2006). The core elements are an activity-based component model, an orchestration model that allows the definition of structured activities, XML schema data types, a service selection model and a mechanism for exception and event handling. Since BPEL is strongly related to Web Services, BPEL4WS is build on top of numerous XML based specifications: WSDL 1.1, XML Schema 1.0 and XPath 1.0. The WSDL descriptions are required for all participating services while the XML Schema specifies the datatypes in conjunction with the WSDL messages and XPath is needed for internal data manipulation.

3 **C**onceptual **D**esign

This chapter designs conceptually the WPS-T approach. The basic approach intends to allow any kind of geoprocess to be deployed at runtime such as executable Jar-files or XML-based workflow descriptions. To foster interoperability, profiles have to be used to machine-readably indicate what kind of process description is understood by the server. Since the dynamic deployment of workflows served as the basic motivation, a BPEL profile will be introduced in this section besides the actual interface specification.

3.1 Operation Specification

As desribed above, the WPS specification has to be extended in order to dynamically deploy and undeploy workflows as shown in Figure 2 as an UML class diagram.

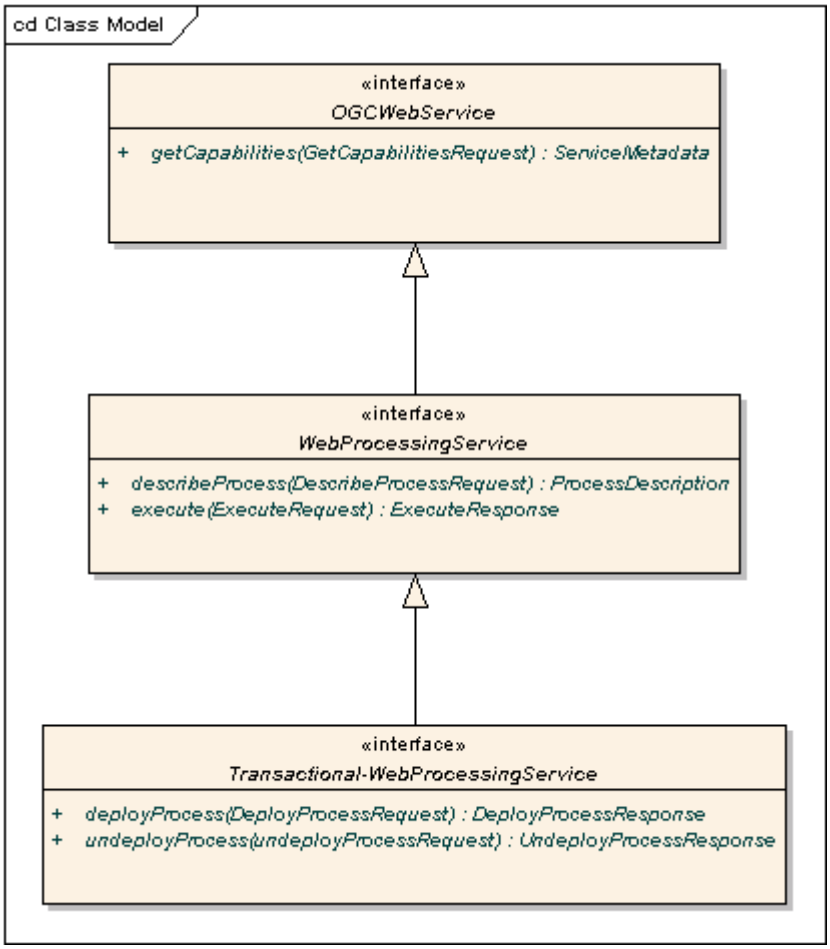

Figure 2: WPS-T Hierarchy.

The Transactional WPS extends the WPS with two new operations: DeployProcess and UndeployProcess. The GetCapabilities is also inherited from the parent WPS which inherits it from the OGCWebService entity. All operations are described in detail in the following sections.

3.1.1 GetCapabilities modifications

The extension of the basic WPS interface with two new operations requires the modification of the GetCapabilities Metadata accordingly. Obviously, both new operations have to be inserted in the <ows:OperationMetadata> tag as a new <Operation> e.g.

```
<ows:Operation name="DeployProcess">
  <ows:DCP>
     <ows:HTTP>
          <ows:Post xlink:href="http://hooters.uni-
              muenster.de:8080/wps/WebProcessingService"/>
     </ows:HTTP>
  </ows:DCP>
</ows:Operation>
```

Listing 1. WPS-T Capabilities Extension.

Secondly, to keep the DeployProcess operation generic, the GetCapabilities metadata has to provide a list of supported schemas profiles for deployment. For instance, if a WPS instance only understands BPEL and not e.g. YAWL or JAR deployment profile, a specific BPEL deployment profile has to be announced in the GetCapabilities metadata. Therefore, the WPS GetCapabilities response schema has to be modified according to listing 1 and 2.

```
<wps:Capabilities>
 […]
    <wps:SupportedDeploymentProfiles>
       <wps:Default>
          <wps:DeploymentSchema
          xlink:href="http://foo.bar/BPEL_Profile.xsd" />
       </wps:Default>
       <wps:Supported>
          <wps: DeploymentSchema
       xlink:href="http://foo.bar/XY_Profile.xsd" />
       </wps:Supported>
    </wps:SupportedDeploymentProfiles>
</wps:Capabilities>
```

Listing 2: WPS-T Deployment Profiles.

This approach allows the WPS to specify any kind of schema which can range from BPEL descriptions to a generic Java source code or executable JARs using a mobile code approach (Vigna 1997) or GRASS function compositions. One default schema has to be set and an optional list of supported schemas. The WPS-T has to understand the schemas and extract the information to expose the described process as a simple process.

3.1.2 DeployProcess

The DeployProcess operation shall offer requestors a means to dynamically deploy a specified process. The XML encoded request shall be send via HTTP-POST and follow the schema specified in figure 3.

3.1.2.1 Request

The DeployProcess Request allows clients to dynamically deploy a process according to the supported deployment profiles (see the previous subsection). Figure 3 presents the structure of a DeployProcess request as a conceptual diagram. After a successful deployment, the process should be

present in the GetCapabilites <wps:ProcessOfferings> list and be accessible like any other process.

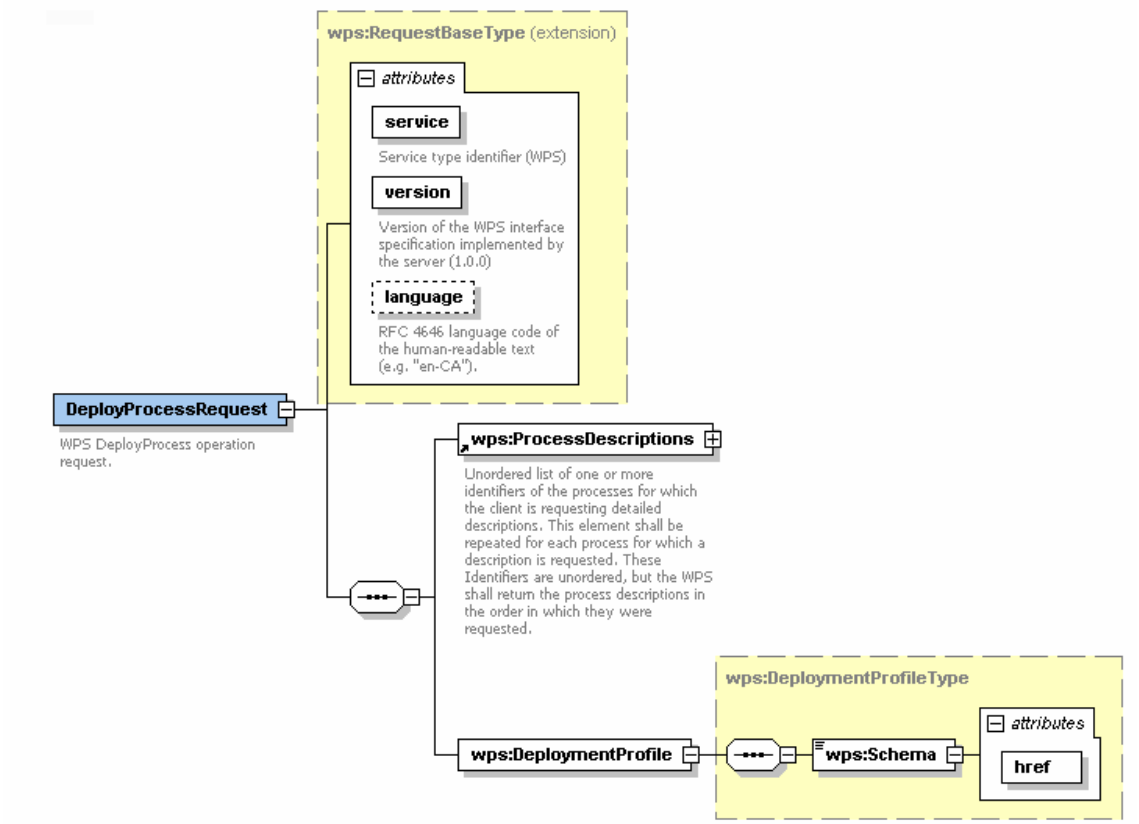

Figure 3: DeployProcess Request Schema.

The <DeployProcessRequest> request element inherits from the general <WPSRequestBase> (see OGC 2007a), which just overrides the service parameter inherited from the <RequestBase> specified in OWS Common (OGC 2007c). This < DeployProcessRequest > itself overrides the inherited request parameter with the CharacterString "*deployProcess*". Additionally it consists of a <wps:ProcessDescription> element as specified by the WPS specification and should contain exactly one process description document as specified by the WPS 1.0.0 specification. This document shall be returned as a result of a DescribedProcess request when this process is deployed. Furthermore, a generic <DeploymentProfile> data structure is part of the deploy process request. This element serves as a toplevel container for any kind of XML encoded process behaviour such as a BPEL script or a referenced or base64 encoded Jar file. Specific profiles have to extend this element to ensure interoperability. Therefore, the <Schema> element of the <DeploymentProfile> element has to match one of the offered deployment schemas in the GetCapabilities metadata which extend the <DeploymentProfile> element.

3.1.2.2 Response

The DeployProcessResponse indicates if the request has succeeded or not. If the process has been successfully deployed and exposed as a simple WPS process, an XML document shall be returned back in compliance with the following schema:

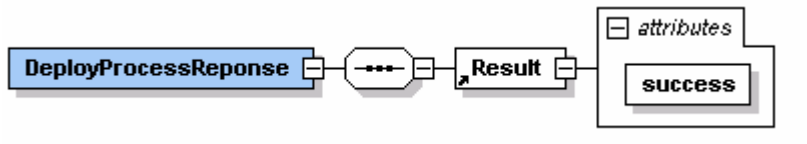

Figure 4: Deploy Process Response.

In case the process could not be successfully deployed an error message should be returned, which is described in the following section.

3.1.2.3 Exceptions

In case a WPS server encounters an error while performing a deployProcess operation, an exception report message as specified in (OGC 2007a) and table 1 will be returned.

Table 1: Deploy Process Exception Codes.

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
oApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
DeploymentProfileNotSupported	The process could no be deployed due to the delivered schema is not supported	Name of operation to be deployed unsuccessfully

3.1.3 UndeployProcess

The UndeployProcess operation offers requestors a mechanism to undeploy a specified process from a WPS. The XML encoded request shall be send via HTTP-POST and shall follow the schema specified in figure 5.

3.1.3.1 Request

The <UndeployProcess> elements should have exactly one <Process> element besides the elements inherited from the <wps:RequestBaseType>. The <Process> element should have an id attribute, which should represent the process id of the process to be undeployed.

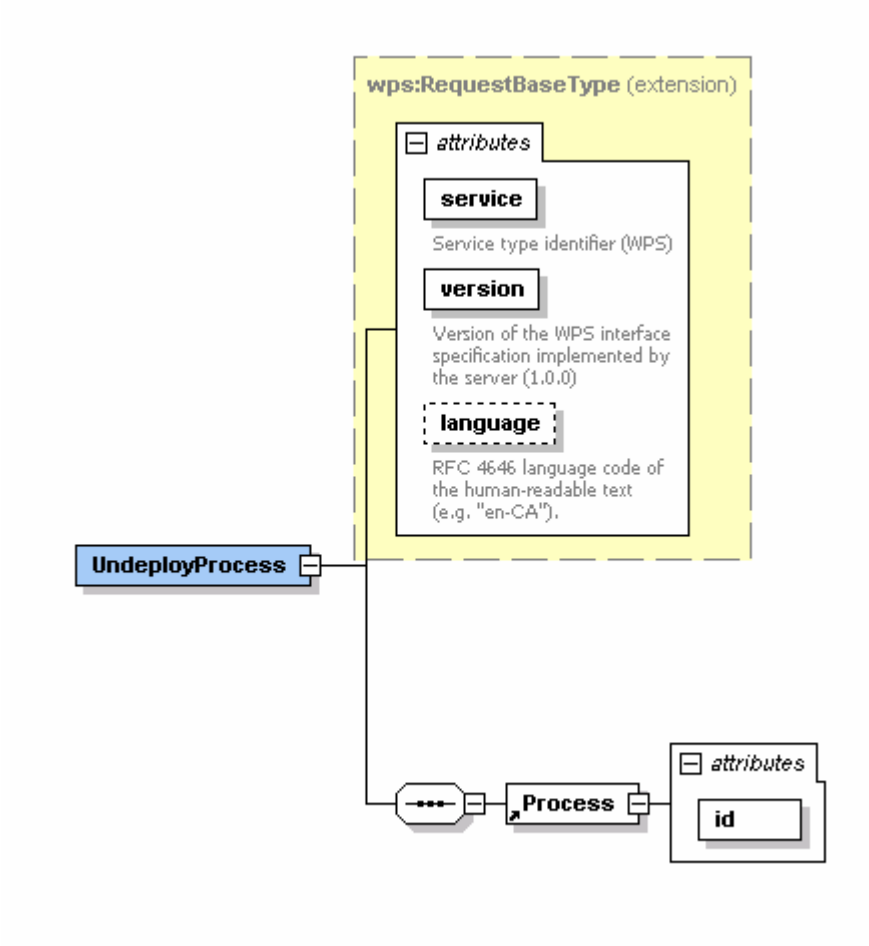

Figure 5. Undeploy Process Request.

3.3.1.2 Response

The response for the UndeployProcess request will be an XML document following the schema indicated in figure 6.

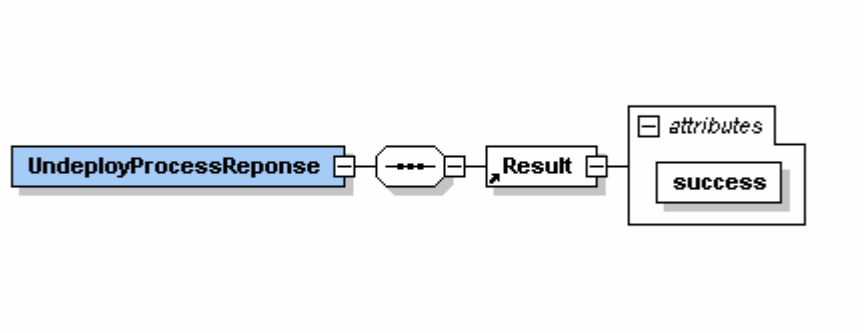

Figure 6: UndeployProcess response schema visualisation.

Analogous to the DeployProcess result, the <Result> element shall indicate via the boolean *success* attribute, if the process could be successfully undeployed. A WPS-T shall be able to undeploy all processes previously deployed via the DeployProcess operation. As not all WPS processes need to be deployed via the WPS-T DeployProcess operation, it is not guaranteed that all process can be undeployed.

3.1.3.3 Exceptions

When a WPS server encounters an error while performing a undeployProcess operation, it shall return an exception report message as specified in (OGC 2007a). and table 2.

Table 2: Undeploy Process Exception codes.

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
UndeploymentFailure	The process could no be undeployed.	Name of operation to be deployed unsuccessfully

3.2 BPEL Profile

As described in section 3.1.1, the WPS-T provides a generic mechanism to support any kind of process description schema. Since the dynamic deployment of workflows was the initial motivation, this study focuses on BPEL as the de-facto workflow description standard (van der Aalst 2003). Therefore, this section introduces a BPEL WPS-T profile. If a WPS offers that schema, clients will be able to expose BPEL workflows as simple WPS processes at runtime. The WPS-T which supports that profile implementation is responsible for either orchestrating or choreographing the workflow or for delegating this task to a third party like a BPEL engine. Figure 7 describes the basic idea. This profile approach keeps the design modular and allows further adjustments and extensions to other requirements.

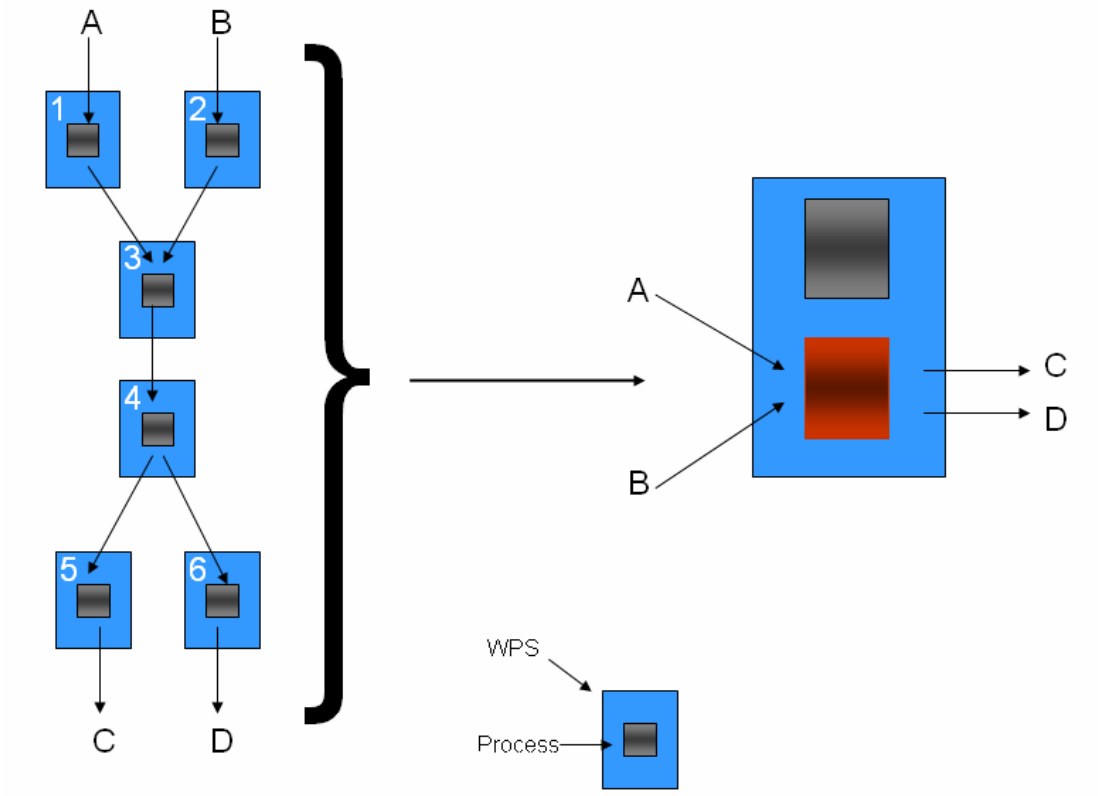

Figure 7: Black Box deployment approach for the Workflow Profile.

The Geoprocessing workflow modeled on the left side of figure 7 consists of blue WPSs each containing a black process. This rather complex process can be seen as a black box, in which only input A and B and output C and D are visible from outside. Thus, this black box is nothing else than a WPS process. Because the workflow can be regarded as a WPS process it could also be executed as any other WPS process, if deployed on a WPS-T. This approach would also enable other workflows to incorporate the modeled workflow in their workflow as a simple WPS process. But to enable the process to be accessible via the WPS standard interface, it has to be deployed onto a WPS using the newly introduced deployProcess operation (see Section 3.1.2), as seen on the right side of figure 7.

This profile inherits from the previously defined <DeploymentProfile> and incorporates all necessary parts as required by BPEL and consists of five mandatory elements as shown in figure 8.

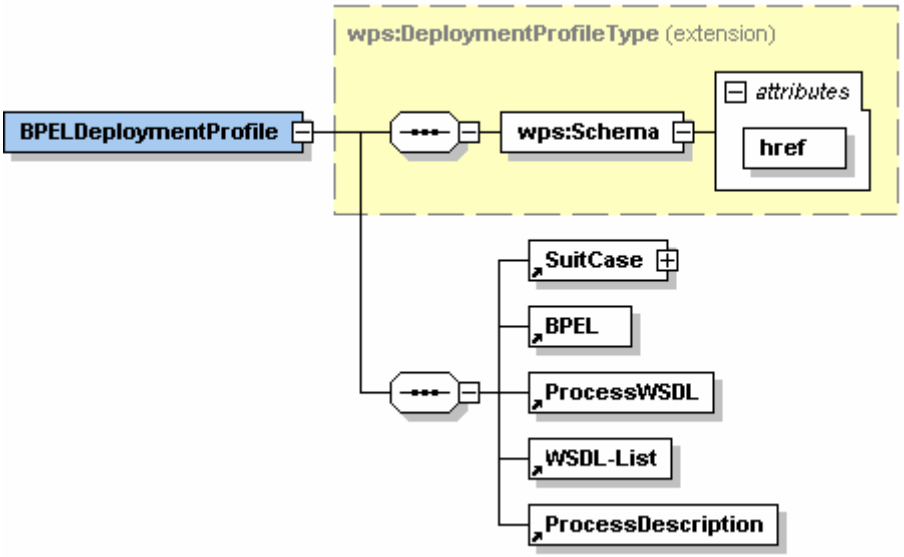

Figure 8: BPELDeploymentProfile Structure.

To insure interoperability of the profiles the <wps:Schema> element is inherited from the <DeploymentProfile> specified in DeployProcessRequest from 3.1.2.1 and indicates the schema for this profile which is referenced in the Capabilities service metadata.

Additionally, the <SuitCase> parameter acts as a meta parameter, which describes the use of other elements. As shown in figure 9, it basically describes the BPEL process to be deployed by the <BPELProcess> element. This element shall have three attributes. The *src* attribute describes the name of the BPEL script. The optional *noAlterWSDL* attribute indicates whether the BPEL engine is allowed to alter the WSDL definition for participating Web Services in the workflow or not. The *id* attribute defines the actual process id under which the process can be invoked.

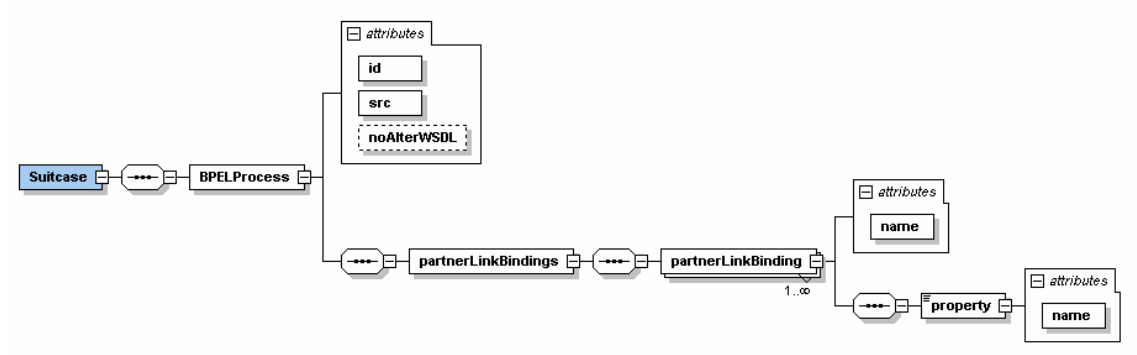

Figure 9: Detailed DeployProcess request schema visualisation

Furthermore, the <BPELProcess> element should have a <partnerLinkBindings> element, which represents a container element for multiple <partnerLinkBinding> elements. Each of these elements

represents a participating Web Service and have a *name* attribute and a <property> element specifying the actual name for the WSDL file for easier deployment on a BPEL engine.

The <BPEL> parameter of the DeployProcess operation shall include the BPEL script representing the modeled workflow. It should follow the BPEL4WS 1.1 specification (Andrews et al. 2003) with the limitation that only BPEL processes with a single receive statement are allowed to avoid the complexity of correlation and multiple WPS processes resulting from one BPEL process.

The <ProcessWSDL> parameter should contain a WSDL file describing the workflow itself applying the WSDL 1.1 schema (W3C 2001), since it can also be seen as a Web Service. The <WSDLList> parameter should include WSDL descriptions for all Web Services participating in the workflow. Each Web Service should be described in WSDL 1.1. Apart from this, the WSDL files should be extended with a <PartnerLink> element since it is required for the BPELEngine.

4. Realization

This chapter picks up the conceptual design and the specification work accomplished in the previous chapter and tries to verify the ideas by a prototypical Java implementation.

The 52°North[3] Web Processing Service implementation was used as the basis for the specified WPS extensions. This implementation is fully compatible to the OGC 1.0.0 standard and provides due to a modularized concept (Schaeffer and Foerster 2007) an ideal starting point.

One goal was to alter the underlying 52°North WPS implementation as little as possible. Therefore, the logic for the two new operations was completely separated and bundled as a new servlet. This technical approach keeps the Transactional WPS extension optional as theoretically intended by section 3. Hence, the DeployProcess and UndeployProcess operations have to be called on a different endpoint than the original operation. This behaviour is supported by the OWS common (OGC, 2007c) specification in terms of allowing different URLs for every operation. In order to not only support predefined algorithms as it was originally implemented by the 52°North implementation, a dynamic concept had to be introduced in order to enable the WPS-T to register and unregister an algorithm at runtime. A static local repository provides access to predefined algorithms, which are available on the same machine as the WPS-T. Dynamic Repositories allow

[3] 52°North website: http://www.52North.org

the registration and unregistration of processes at runtime. The BPELRepository in figure 10 is a sample implementation which implements the BPEL Profile specified in section 3.2 and acts as a dynamic wrapper for a BPEL engine.

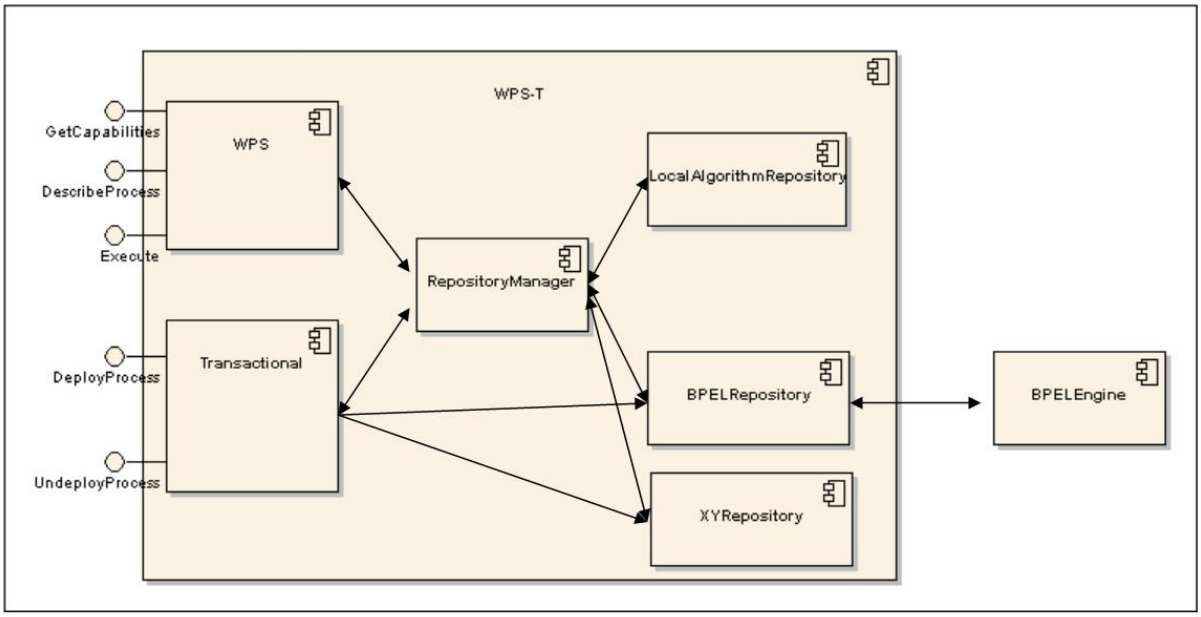

Figure 10: Basic WPS-T Architecture.

Workflows deployed on a BPEL engine are accessed and wrapped as algorithms by this repository. The WPS-T servlet directly accesses the BPEL repository for deploying/undeploying processes which follows the BPEL deployment profile specified in section 3. Since the WPS servlet is responsible for executing the processes, it requests the repository manager to get access to all registered algorithms. By encapsulating algorithms from static and dynamic algorithm repositories behind this unified interface, the WPS servlet does not need to know what kind of repositories and algorithms are registered. It only needs a standardized way of fetching the necessary process which is delivered by the RepositoryManager. Therefore, the architecture is held open to support different repositories supporting different deployment profiles as indiacated by the XYRepository in figure 10.

The following sections describe the taken approach in detail.

4.1 A generic repository concept

A new repository concept was introduced to the 52°North WPS core functionality. This became necessary because the new WPS-T technology requires a dynamic repository approach as described above. The WPS

concept of loading processes classes from a properties file was replaced by a generic repository concept. As shown in figure 11, all repositories are registered at startup at the org.n52.wps.server.RepositoryManager. Other classes from the WPS business logic are able to obtain an instance of a class implementing org.n52.wps.server.IAlgorithm interface via the getAlgorithm(String algorithmID) method.

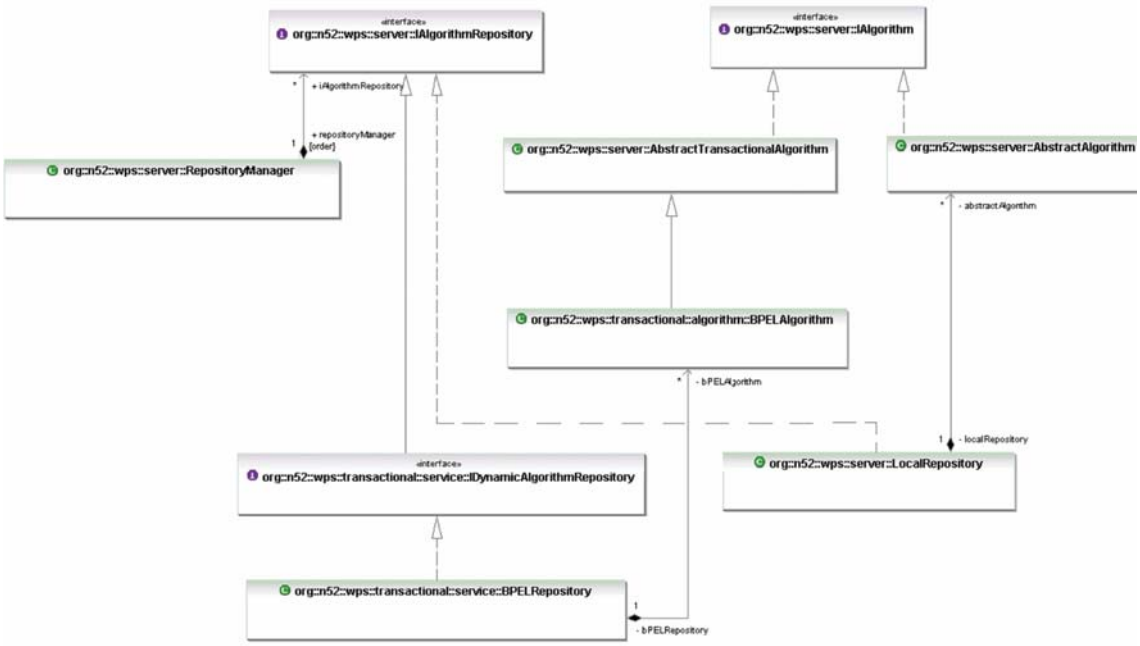

Figure 11: WPS-T architecture as a UML class diagram

This approach follows in principle the factory pattern (Gamma et al. 2005) because specific repositories have to implement the org.n52.wps.server.IAlgorithmRepository interface and use their own behaviour of creating algorithms. The RepositoryManager.getAlgorithm(String algorithmID) method searches in every registered repository for the requested algorithm. The RepositoryManager.getAlgorithms() method returns a list of all currently registered algorithm ids. To obtain all algorithms ids, every registered repository is requested for their registered algorithm ids by calling the getAlgorithmNames() method.

This flexible approach enables the WPS-T to register multiple repositories and delegates the responsibility for fulfilling the contract manifested by the IAlgorithmRepository and IAlgorithm interface in the hands of the actual implementations. Therefore, this concept allows the WPS-T to work with static and dynamic repository behaviors.

4.2 Static Repositories

The WPS-T implementation is equipped with two different repository types. The default repository is the org.n52.wps.server.LocalAlgorithmRepository. It is a static repository because at start-up this repository reads a properties file once and creates an instance for every registered algorithm derived from the abstract org.n52.wps.server.AbstractAlgorithm class. The created classes are stored in a hashtable identified by their classname. Additionally, the abstract org.n52.wps.server.AbstractAlgorithm class provides an already implemented method for creating the describeProcess document.

4.3 Dynamic Repositories

The new Dynamic Repository approach has three levels of abstraction. On the general level, it defines the IDynamicAlgorithmRepository interface, which extends the basic IAlgorithmRepository. So, on the one hand, it can still be requested by the RepositoryManager in order to retrieve registered algorithm by the IAlgorithm-Repository methods. On the other hand, the IDynamicAlgorithmRepository requires dynamic behavior by specifying the IDynamicAlgorithmRepository.addAlgorithm() and IDynamicAlgorithmRepository.removeAlgorithm() methods. This approach allows different kinds of dynamic repositories to be registered. On the second level of abstraction, a particular dynamic repository for BPEL processes was implemented as a sample dynamic repository using the abstract repository concept. The BPELRepository class, which implements the IDynamicAlgorithmRepository interface and uses an implementation of the IDeployManager interface for the actual communication with the BPEL engine. The IDeployManager is the third level of abstraction, since it allows the use of different kinds of BPEL engine. For instance, the OracleBEPLManager class can be substituted by a class connecting to an ActiveEndpoints[4] BPEL engine, without changing the toplevel classes, since they all operate on interface methods.

The implemented repository mechanisms are validated by a real world use case described in the following section.

5. A Distributed Processing Scenario on Air Quality

European Union legislation (Council Directive 1999) brought air quality into the spot light. The legislation has to be adopted by national law by all member states and sets strict limits for different air quality parameters. This

[4] http://www.active-endpoints.com/active-bpel-engine-overview.htm

use case focuses only on Particulate Matter (PM10) as one of the most hazardous aerosols (BMFU 2005).

In order to maintain the specific limits on PM10, urban agglomeration areas have addressed that issue in different ways (Kossak 2004). Like most urban agglomeration areas, several parts of Germany exceed the PM10 limits on a regular basis (DIN 2006). Therefore, the temporal closing of highly frequented roads for vessels with diesel engines can be one solution in order to maintain the EU limits, since the amount of traffic induced PM10 is around 40% in urban areas (CAFE 2004).

As a proof of concept of the introduced approach in this study, a generic model is developed, which indicates road sections in highly polluted areas based on interoperable Web Services. The distributed data and services are incorporated in a Geoprocessing Workflow to form the model and produce the desired output. As input data for this model, road data of North-Rhine-Westphalia from an ATKIS[5] database delivered through an Interactive Instruments[6] WFS 1.1.0 hosted by the LDS[7] is provided as well as close-real-time air quality data collected from seventy-one monitoring stations representatively spread over NRW and delivered through a Sensor Observation Service wrapped by a WPS.

To solve the given problem, a workflow has to be modelled consisting of five loosely coupled WPS. For demonstration purpose, the whole workflow will be split into a basic workflow and an advanced workflow which will incorporate the basic one.
As presented in figure 12, the basic workflow takes as input a SOS URL and outputs interpolated polygons based on the input data with the help of three service. The first service encapsulates a SOS and outputs O&M which only contains PM10 measurements. The second service takes the O&M document from the first service and transforms it into GML point features. The final service in this basic workflow interpolates GML point features via the Thiessen Interpolation method and outputs GML polygons. After modelling this basic workflow, it has to be deployed via the WPS-T interface to be accessible and reusable over the web as a simple WPS process. Thus, a WPS-T deployProcess request has to be formed and all necessary data has to be acquired. Since we want to use a BPEL profile, the request has to follow the schema presented in section 3.2. Fortunately, the

[5] http://www.atkis.de/
[6] http://www.interactive-instruments.de/index.php?id=1&L=1
[7] The WFS URL is not published here due to security reasons.

used 52°North WPS-Workflow modeller (Schaeffer 2007) supports the automatic creation of such a request and it does not become necessary to construct WSDLs, describeProcess and deployment descriptor documents manually. The WPS-T deployProcess operation is successfully invoked with the request describing the modelled workflow and the process is made available under the identifier: org.n52.wps.algorithm.sample.SOSInterpolation.

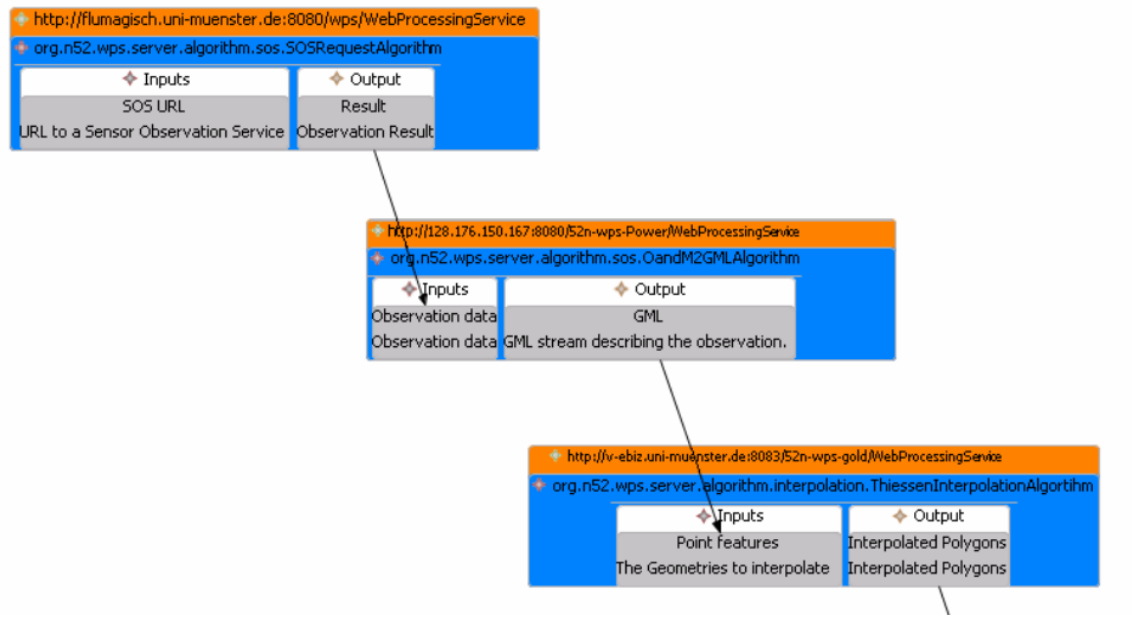

Figure 12: Basic Model.

In order to finalize the give task, it is necessary to calculate road sections in highly polluted areas. Therefore, a second workflow is modelled, which takes the org.n52.wps.algorithm.sample.SOSInterpolation WPS process as first element in the workflow and thereby incorporates the basic workflow modelled above. Since the basic workflow is exposed as a simple WPS process, there is no distinction between this process and any other offered process. The second process takes GML polygons as input and filters attributes over a given threshold value. The output contains polygons which have the specified attribute above the given threshold. As the final step, these filtered polygons are intersected with road data delivered as GML via a WFS. As a result, only those road sections inside of a filtered polygon are returned. These road sections are the road sections in highly polluted areas and therefore the composite workflow provides a means to generically solve the given task for any given area.

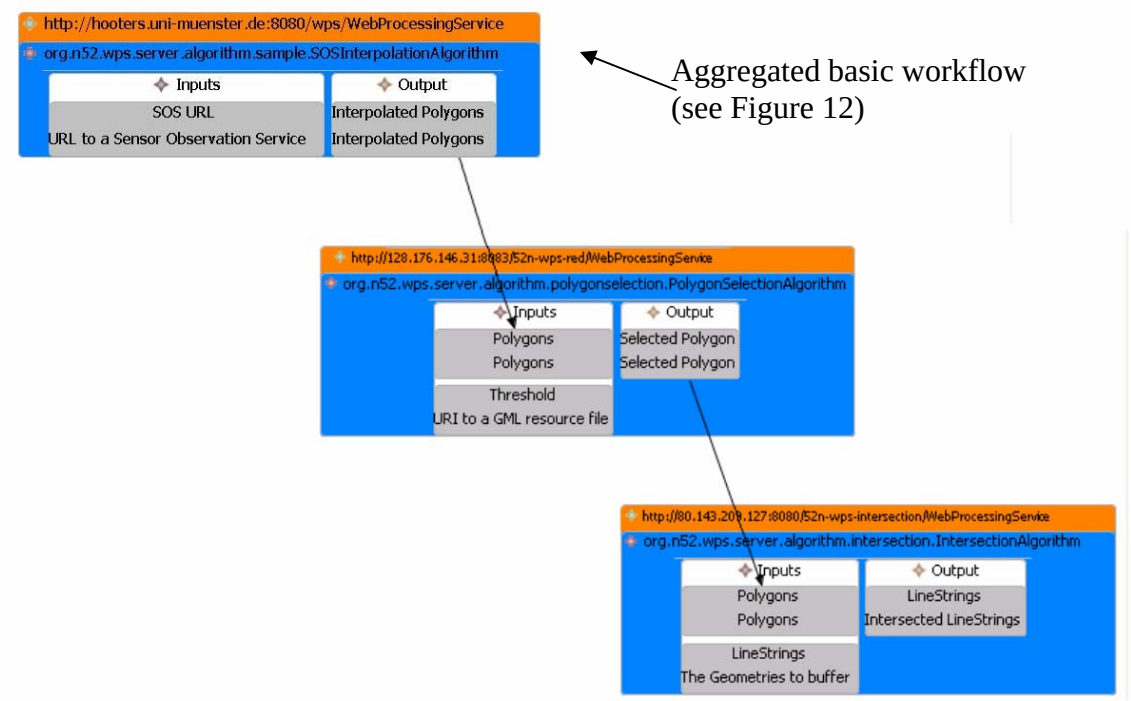

Figure 13: Advanced Workflow structure.

This final workflow is now ready to be deployed on WPS via the deployProcess operation analogous to the procedure described for the basic workflow. Since it is now available as a simple WPS process it can be executed like any other process. The WPS client developed by 52°North (Foerster and Schaeffer 2007) provides a means to execute any WPS process in an easy manner and since it is integrated in uDig, the results can be visually explored. Figure 14 presents the results as the red colored roads on the east side of the map visualized in the uDig client.

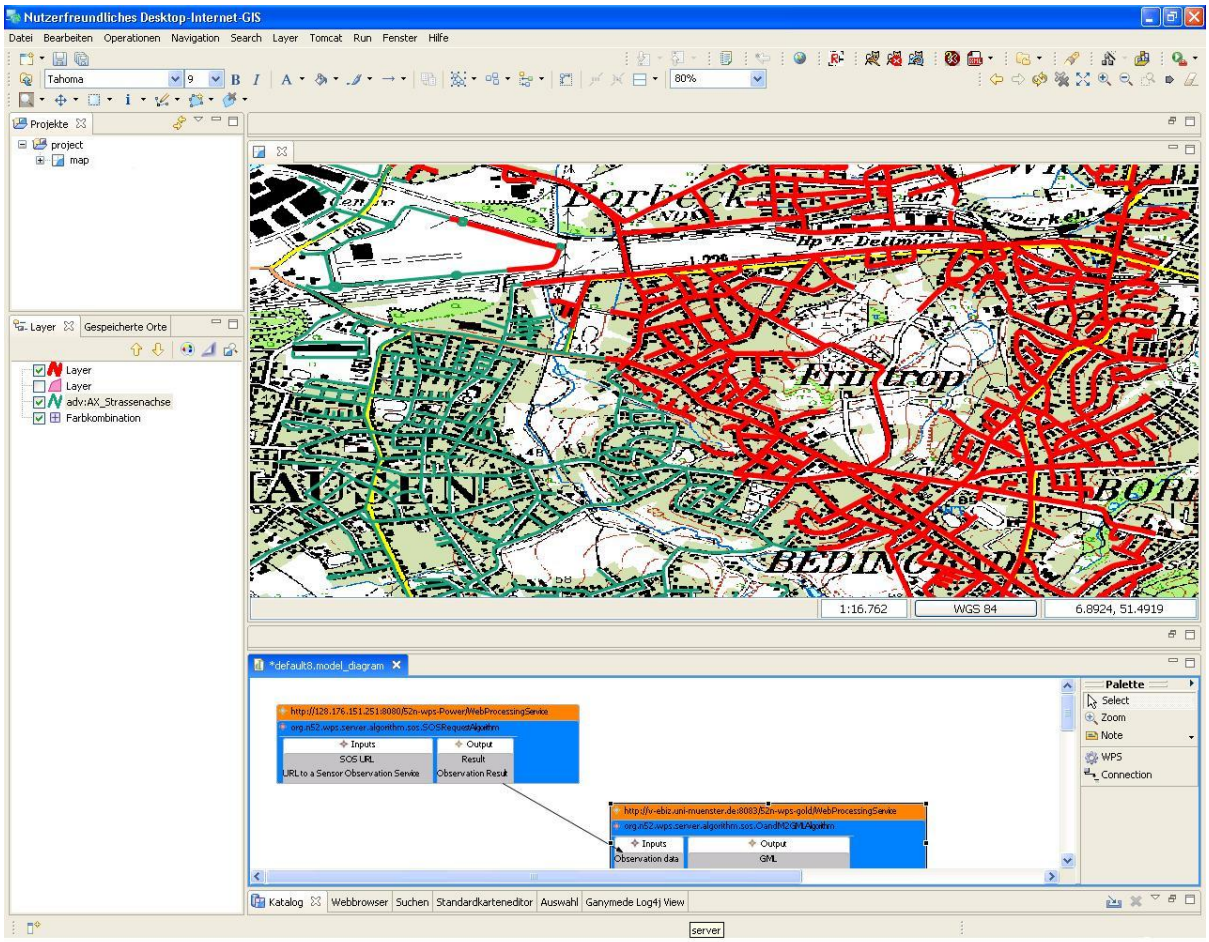

Figure 14: Workflow results (red road layer on the east side).

6. Outlook & Conclusion

The introduced Transactional Web Processing Service approach provides a highly flexible means to dynamically deploy and undeploy WPS processes since it can be defined as an extension to the existing WPS specification. With the specified ability to provide a list of supported schemas in the service description delivered through the GetCapabilities operation, the newly introduced operations are not limited to any kind of schema or deployment information and thus foster reusability and flexibility but still maintaining interoperability by means of the introduced schema inheritance. Additionally, this approach blends in the already existing WPS specification which is also held open in terms of supported schemas and processing functionality.

The specified BPEL deployment profile as one possible deployment schema peculiarity in conjunction with the introduced WPS-T was successfully validated by the presented use case. By providing a fairly deep

insight into the implementation details and the three levels of abstraction, it was explained how the presented implementation of the BPEL profile can deal with different BPELEngines in the backend and therefore supports the easy adoption of changing external conditions.

Reusability of modelled workflows and seamless integration were focused as the strength of the proposed approach. However, BPEL is only one possibility and comes along with some challenges for the OGC world by (Weiser & Zipf 2007) which could be solved in a convenient manner by the specialized 52°North WPS modeller uDig plugin. Nevertheless, e.g. WSDL descriptions can be created manually and also used as input along with a BPEL script. In other words, the proposed extension does not rely on the 52°North WPS modeller but it provides a convenient means to bridge the gab between the mainstream IT and the OGC world and automatically request the WPS-T.

The applied approach of opaque chaining (Alameh 2003) for the BPEL profile also indicated an up-to-date unsolved issue of given the client/user a means to explore the type of offered WPS process and thus transforming the WPS-T opaque process pattern into a transparent one. Even if the user is only interested in the results of a process, transparent processes might help in conjunction with semantic annotation to better choose an appropriate process and understand the results.

Nevertheless, the presented approach closed the gap of dynamically deploying processes in a generic but standardized way. This enriched the WPS idea and prepared it for a broader area of application especially in the geoprocessing workflow world and thereby for future requirements.

References

Alameh, N. (2003). “Chaining geographic information Web Services”. IEEE Internet Computing, 7 (5): 22-29.

Alonso, G., Casati, F. and Kuno, H. (2003). Web Services Concepts, Architectures and Application. Berlin, Springer.

Andrews, T., Cubera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D. Smith, D., Thatte, S., Trickovic, I. and Veerawarana. S. (2003). Business process execution language for Web Services [online]. Available from: http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf [Accessed on January 14th 2008].

Bernard, L., Crompvoets, J. and Fritzke, J. (2005): Geodateninfrastrukturen – ein Überblick. Wichmann, Heidelberg.

Umwelt Bundesamt (BMFU) (2005). Hintergrundpapier zum Thema Staub/Feinstaub (PM), Berlin.

CAFÉ Working Group on Particulate Matter(Ed.) (2006). Second Position Paper on Particulate Matter. [online] http://ec.europa.eu/environment/air/cafe/pdf/working_groups/2nd_position_paper_pm.pdf [Accessed on January 14th 2008].

Cepicky, J. (2006). Grass goes web: PyWPS. Free and Open Source Software for Geoinformatics, Lausanne, Switzerland (11-15 September 2006)

Chen, L., Wassermann, B., Emmerich, W. and Foster, H. (2006). Web Service Orchestration with BPEL [online]. London Software Systems, Dept. of Computer Science, University College London. Available from: http://sse.cs.ucl.ac.uk/omiibpel/publications/tut15-emmerich.pdf [Accessed on January 14th 2008].

Council Directive 1999/30/EC of the European Parliament and of the Council (1999). Official Journal of the European Union, 22. April 1999, pp. 1-21.

Deutsches Institut für Normung e. V(Ed.) (2006). Feinstaub und Stickstoffdioxid. Beuth , Berlin.

Ernst, H. (2000). Grundlagen und Konzepte der Informatik. Eine Einführung in die Informatik ausgehend von den fundamentalen Grundlagen. Braunschweig, Vieweg.

Foerster, T. (2006). An open software framework for web service-based geo-processes. Free and Open Source Software for Geoinformatics, Lausanne, Switzerland (11-15 September 2006)

Foerster, T. and Stoter, J. (2006). Establishing an OGC web processing service for generalization processes. ICA workshop on Generalization and Multiple Representation, Portland (25 June 2006).

Foerster, T., Schäffer, B. (2007). “A Client for Distributed Geo-processing on the Web”. Lecture Notes in Computer Science (LNCS), vol. 4857, 7th International Symposium on Web and Wireless GIS (W2GIS 2007), 252-263.

Gamma, E., Helm, G., Johnson, R. and Vlissides, J. (2005). Design Patterns. Reading, Addison-Wesley.

Gehlot, S. and Verbree, E. (2006). Web-based sharing of a Geo-processing chain: combination and dissemination of data and services. ISPRS IV Geospatial databases for sustainable development, Goa.

Gottschalk, K., Graham, S., Kreger, H. and Snell, J. (2002). “Introduction to web service architecture”. IBM Journal 41(2): 170–177.

Groot, R., McLaughlin, J. (2000). Geospatial data infrastructure: concepts, cases and good practice. New York, Oxford University Press.

Keens, S. (2007). OWS-4 Workflow IPR. OGC discussion paper, OGC 06-187r1.

Khalaf, R. and Leymann, F. (2003). “On Web Services Aggregation”. B. Benatallah, and M.C. Shan, eds. *LNCS 2819*, Berlin Heidelberg: Springer, 1–13.

Kiehle, C. (2006.) “Business logic for geoprocessing of distributed data”. Computers & Geosciences 32(10): 1746-1757.

Kiehle C., Greve K., and Heier C. (2006). “Standardized Geoprocessing - Taking Spatial Data Infrastructures one Step Further”. Proceedings AGILE 2006, 273-282.

Kossak, A. (2004). “Straßenbenutzungsgebühren weltweit“. Internationales Verkehrswesen, No. 56, 246-249.

Masser, I. (2005): GIS Worlds – Creating Spatial Data Infrastructures, ESRI Press, Redlands

OGC (2007a). OpenGIS Web Processing Service. OGC Implementation Specification OGC 05-007r7, Open Geospatial Consortium.

OGC (2007b). *OpenGIS* Geography Markup Language. OGC implementation specification, OGC 07-036, Open Geospatial Consortium.

OGC (2007c): *OpenGIS* Web Services Common Specification. OGC implementation specification, OGC 06-121r3, Open Geospatial Consortium.

ORCHESTRA (2007). “Service Chain Access Service Specification 0.4”. Available from: http://www.eu-orchestra.org/download.php?file=docs/OA-Specs/Service_Chain_Access_Service_Specification_v0.4-JRC-IES.pdf [Accessed on May 4th 2008]

Pelz, C. (2003a). “Web services orchestration and choreography”. Computer , 36 (10), 46-52.

Pelz, C. (2003b). Web services orchestration: A review of emerging technologies, tools, and Standards, Technical report, Available from: http://xml.coverpages.org/HP-WSOrchestration.pdf [Accessed 14th January 2008].

Reichert, M. and Stoll, D. (2004). “Komposition,Choreograhpie und Orchestrierung von Web Services – Ein Überblick“. *EMISA* Forum, 24(2) 21-32.

Schaeffer, B. (2007). “Integrated Web Geoprocessing Workflow Composition and Deployment”, Diploma thesis, University of Muenster, Muenster. Available from: http://v-ebiz.uni-muenster.de:8083/Schaeffer/DA/Integrated%20Web%20Geoprocessing%20Workflow%20Composition%20and%20Deployment%20-%20Schaeffer.pdf [Accessed on May 4th 2008].

Schaeffer, B. and Foerster, T. (2007). Bringing the Web Processing Service to a new stage – new 52°North WPS Features. . Free and Open Source Software for Geoinformatics, Victoria, Canada (24-27 September 2007)

van der Aalst, W.M.P (2003). „Don’t go with the flow: Web services composition standards exposed”. IEEE Intelligent Systems, 01 (02) 72-76.

van der Aalst, W M. P., Dumas, M., ter Hofstede, A., Russell, N., Verbeek, H., and Wohed, P. (2005). “Life After BPEL?” Proc. of the 2nd Int. Workshop on Web Services and FormalMethods (WS-FM), LNCS 3670, 35–50.

Veerawarana, S., Curbera, F., Leymann, F., Storey, T., Fergusaon, D.F. (2005). Web Services Platform Architecture: SOAP, WSDL, WSPolicy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More. Englewood Cliffs, Prentice Hall.

Vigna, G. (1997). "Mobile Code Technologies, Paradigms, and Applications", PhD thesis, Politecnico di Milano, Milano.

W3C (2001). Web Service Description Language (WSDL) 1.1. [online] W3C, Online: http://www.w3.org/TR/wsdl/ [Accessed on January 14th 2008].

Weiser, A. and Zipf, A. (2007). “Web Service Orchestration of OGC Web Services for Disaster Management”. J. Li, S. Zlatanova and A.G. Fabbri ,eds. Lecture Notes in Geoinformation and Cartography Geomatics Solutions for Disaster Management, Berlin Heidelberg, Springer, 239-254.

Weiser, A., Neis, P. and Zipf, A. (2006). “Orchestrierung von OGC Web Diensten im Katastrophenmanagement - am Beispiel eines Emergency Route Service auf Basis der OpenLS Spezifikation“. GIS-Business 9, 35-41.

Wohed, P., van der Aalst, W.M.P., Dumas, M. and ter Hofstede, A.H.M. (2003). Analysis of Web Services