GEOSPATIAL WEB SERVICES FOR DISTRIBUTED PROCESSING - APPLICATIONS AND SCENARIOS

Theodor Foerster[1], Bastian Schäffer[1], Bastian Baranski[1] & Johannes Brauner[2]

[1]{theodor.foerster;schaeffer;baranski}@uni-muenster.de
Institute for Geoinformatics, University of Münster, Germany

[2]johannes.brauner@tu-dresden.de
Geoinformation Systems, Technische Universität Dresden, Germany

Abstract. Processing and modeling of geodata are an essential part of the daily work of GIS technology experts. Domain experts often need to perform sophisticated GIS analysis of complex data. Currently, capturing, storing and requesting data are embedded in Spatial Data Infrastructures using Service-Oriented Architectures. GIS analysis is performed locally by first downloading geodata such as from SDIs. With the advancements in network bandwidth, processing power and the standardization of Web technology and Geospatial Web Services, distributed geoprocessing is the next step of realizing GIS analysis on the Web. Geoprocessing Services are considered to be a key aspect of meeting the requirements for distributed geoprocessing on the web.

This chapter provides an overview of the current state-of-the-art approach of distributed geoprocessing by describing the related concepts, such as the OGC Web Processing Service, workflows, Quality of Service and legacy system integration. Furthermore, the chapter demonstrates different applications for distributed geoprocessing. Finally, this chapter examines the introduced concepts by two scenarios.

1 Introduction

Geospatial data and maps have become increasingly accessible remotely in a distributed fashion through standardized Geospatial Web Services using Service-Oriented Architectures (SOA) (Lake & Farley, 2007). SOAs are a technical foundation for implementing Spatial Data Infrastructures (SDIs). SDIs focus especially on providing access to and sharing of geospatial data by defining the technical and organizational framework. The technical framework is often based on common data models and a set of standard Geospatial Web Service interfaces integrated using the SOA paradigm. A number of Geopatial Web Service interfaces are specified as open standards by the Open Geospatial Consortium (OGC) and the International Organization for Standardization (ISO). Currently SDIs address data delivery and web mapping, but the requirement for real-time geoinformation (extracted by GIS analysis based on most recent geodata in real-time) is not yet met by current SDI implementations. The need for near real-time geodata and geoprocessing increases for many geographic applications, such as emergency services, risk management and alerting, in which data from different sources have to be integrated to support decision making with real-time geoinformation. Geoinformation is generated from geodata using geoprocess models (i.e. models of real-world geoprocesses). Thus, embedding geoprocess models into the web, which facilitate distributed Geospatial Web Services for providing geospatial data in real-time, is a prerequisite to achieve web-based geoinformation. In this chapter, creating and performing geoprocess

models that are encapsulated as Web Services and access remote resources (i.e. Geoprocessing Services or Data Services) to generate geoinformation is called *distributed geoprocessing*.

The aim of this chapter is to give an overview about the current approaches for distributed geoprocessing. The chapter will introduce the basic concepts (Section 2) for distributed geoprocessing by using the OGC Web Processing Service (WPS), establishing geoprocessing workflows, ensuring Quality of Service (Grid and Cloud Computing) and integrating functionality of legacy systems. Based on these concepts the chapter describes applications (Section 3), in which distributed geoprocessing on the web is utilized to provide real-time geoinformation. In particular, it will demonstrate how geoinformation extracted from web-based geospatial data can be integrated into mass-market applications. Two scenarios are described based on the introduced concepts to illustrate the full potential of distributed geoprocessing on the Web (Section 4). Section 5 will conclude the described approaches and will provide an outlook about future challenges. All the examples mentioned in this chapter are implemented as Open Source software at the 52°North Geoprocessing Community[1].

As this chapter focuses on distributed geoprocessing, the reader may refer to other chapters of the book for getting specific information on related topics such as Geospatial Web Services, data services (e.g. WFS, WCS).

2 Relevant Concepts for Distributed Geoprocessing

Geoprocesses are real-world processes that are modeled in computer systems to simulate and analyze real-world phenomena. For this chapter geoprocessing is defined as the application of a model representing a real-world geoprocess. As a result, geoprocessing is the transformation of geodata to geoinformation. The definitions of geodata and geoinformation are closely related to the definitions of data and information besides their specific geospatial focus. Geodata and geoinformation describe geospatial phenomena at different levels of abstraction. The terms data and information are not clearly defined in literature. This chapter follows the definitions of Ackoff (1989) and Chen et al. (2009). Ackoff (1989) defines data as a set of symbols and information as data which is processed to answer specific questions. Chen et al. (2009) applied these definitions to the computational domain by defining data as computational representations of models and attributes of real world or simulated entities (i.e. geospatial phenomena). Whereas information is data which have meaning attached and is thereby understandable by computational systems or human users.

Consequently, a geoprocess model handles geodata in a geospatial context. The input and the output contain geodata or its interpretation (e.g. a Boolean value to answer, if two geometries intersect each other) and the applied calculation takes the geospatial context into account. Examples of such models are simple buffer computations, geostatistical analysis or large-scale simulations of noise distribution.

A Web Service can be defined as a software component that provides functionality including access to data sources through a web-accessible interface in a programming language- and platform-independent manner (Vaughan-Nichols, 2002). The Web Service interface is described in a machine-understandable way, which is a fundamental requirement for interoperability. Based on these interfaces Web Services connect readily available software components on the Web in a loosely coupled way (Alonso, Casati, Kuno, & Machiraju, 2004). Loosely coupled means that the service interaction is established during runtime and the services do not know each other in advance. This enables to reuse software components

[1] 52°North Geoprocessing Community website: www.52north.org/wps.

in different applications. Moreover, as Web Services communicate based on platform-independent protocols (i.e. Hypertext Transfer Protocol) and exchange formats (i.e. XML), they can be reused by any application written in any programming language and/or running on any operating system. Additionally, Web Services are stateless software components that do not expose a specific state to the client and remain stateless before and after client interaction. This is a central design characteristic to keep the architecture scalable and flexible for many different applications.

Establishing and performing geoprocess models over a network was initially achieved in the 1980s, when clients were able to trigger computations performed on remote computers. Since the advent of the Web and Web Services, it is possible to establish distributed architectures. Distributed architectures involve more than two remote nodes communicating with each other in a loosely-coupled way. This loosely-coupled communication is a characteristic of the web and is reflected for instance by the Publish-Find-Bind paradigm of SOA, in which the binding between the different nodes is established during runtime (Alonso et al., 2004).

Consequently, an architecture is termed distributed if it consists of more than two entities not located on the same node. Thereby several mechanisms are required to establish distributed geoprocessing:

- Standardization of the single nodes (i.e. Data Services, Registries and Geoprocessing Services)
- Managing the communication between multiple nodes through geoprocessing workflows
- Ensuring a Quality of Service (operational level)
- Incorporating existing geoprocess models (functional level).

Establishing single (OGC WPS) and complex geoprocess models (geoprocessing workflows) is the foundation for realizing distributed geoprocessing. Further, distributed geoprocessing can be enhanced once these geoprocess models are established on an operational level and on a functional level.

A requirement for establishing distributed architectures is interoperability. Interoperability is defined as the capability of two services to communicate at runtime to meet a common goal (Alonso et al., 2004). Web Services ensure interoperability by common service interfaces and standardized message encodings.

Interoperability can be established on two levels: the syntactic level and the semantic level. Syntactic interoperability is ensured by common service interfaces (i.e. syntax of input and output parameters). Semantic interoperability is established by meaningful interpretation of the interfaces and the exchanged messages.

One of the research challenges for Geoprocessing Services is semantic descriptions, which are the means to enable semantic interoperability of Web Services (Brauner, Foerster, Bastian Schaeffer, & Baranski, 2009). Other research challenges are:

- Service orchestration
- Strategies to improve performance.

This section will introduce the OGC Web Processing Service (OGC WPS) as the building block for establishing geoprocess models on the Web. Based on OGC's specification baseline, the WPS defines a way of publishing and performing geoprocess models on the web. As geoinformation is typically generated using some geoprocess models that are composed of single Geoprocessing Services, geoprocessing workflows play a major role (Section 2.2). In this respect this section also introduces an OGC compliant way of publishing workflows on the web based on the WPS interface specification

(WPS-Transactional). To improve distributed geoprocessing on the operational level in terms of reliability and performance, Grid and Cloud Computing and corresponding Quality of Service (QoS) guarantees will be described in Section 2.3. Finally, to enable the reuse of existing GIS functionality (incorporated in so-called legacy systems) and improve the functional level of distributed geoprocessing, Section 2.4 demonstrates an approach for wrapping existing functionaliy of legacy systems with a standard interface for distributed geoprocessing. Further usage of existing functionality contributes to the sustainability of SDIs.

2.1 OGC Web Processing Service

In 2007, the OGC approved version 1.0.0 of the OGC Web Processing Service interface specification. The WPS interface specification describes a standardized method to publish and execute web-based geoprocess models of any type (OGC, 2007). According to the WPS interface specification, a geoprocess model is defined as any calculation operating on geodata.

In detail, the WPS interface specification describes three operations, which are all handled in a stateless manner: GetCapabilities, DescribeProcess and Execute. GetCapabilities is common to any type of OGC Web Service and returns service metadata. In case of WPS it also returns a brief description of the geoprocess models offered by the specific service instance. To get more information about the hosted geoprocess models, the WPS provides metadata about the geoprocess model through the DescribeProcess operation. This operation describes all input and output parameters of the geoprocess model. Based on this information the client can perform the Execute operation with the specific input parameters upon the designated geoprocess model. This course of action is depicted in Figure 1. Section 2.1.1 illustrates this course of action by request examples performing desired generalization functionality. Section 2.1.2 describes how semantic interoperability is currently addressed by the WPS interface specification.

For the remainder of this chapter, geoprocess models provided by WPS are called *WPS-based geoprocess models*.

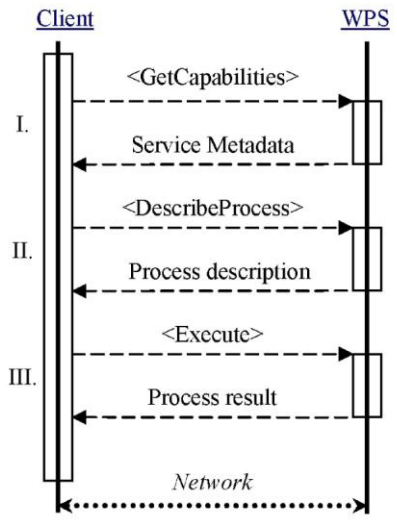


Figure 1: Basic WPS communication.

2.1.1 OGC WPS in Action

To illustrate a workflow sequence that triggers a WPS-based geoprocess model, this section describes the basic communication. In this example, a WPS instance is queried for a Douglas-Peucker algorithm (Douglas & Peucker, 1973) to simplify a set of road geometries. The listed XML messages are in most cases generated by client applications, which guide the user during Web Services interaction by harvesting user input and performing the requests accordingly. Such a client application is presented in Section 3.3. The listed examples are all based on the current version 1.0.0 of the WPS interface specification (OGC, 2007).

For reasons of simplicity it is assumed that the user knows the entry point of the WPS instance in advance. In real-world scenarios such entry points can be retrieved from catalog services. To get more information about the service the user queries the service metadata using a GetCapabilities request (Listing 1). The response of the WPS instance is depicted in Listing 2. From this response document the user can retrieve the service metadata, such as entry points for further communication (`OperationsMetadata`) or individual information about the provider (`ServiceProvider`). Also the provided geoprocess models are listed in the (`ProcessOfferings`). One of these geoprocess models listed is the Douglas-Peucker algorithm, which is briefly described with identifier, title and abstract. The identifier of the geoprocess model can be used to retrieve further metadata.

Listing 1: Example GetCapabilities request for WPS.

```
http://geoserver.itc.nl:8080/wps/WebProcessingService?REQUEST=GetCapabilities&Service=WPS
```

Listing 2: Example GetCapabilities response for WPS.

```
 <?xml version="1.0" encoding="UTF-8"?>
<wps:Capabilities service="WPS" version="1.0.0" xml:lang="en-US"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://geoserver.itc.nl:8080/wps/schemas/wps/1.0.0/wpsGetCapabilities_response.xsd" updateSequence="1"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ows:ServiceIdentification>
    <ows:Title>My WPS</ows:Title>
    <ows:Abstract>Service based on the 52north implementation of WPS 1.0.0</ows:Abstract>
    <ows:Keywords>
      <ows:Keyword>generalization</ows:Keyword>
      <ows:Keyword>geoprocessing</ows:Keyword>
    </ows:Keywords>
    <ows:ServiceType>WPS</ows:ServiceType>
    <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
    ...
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
    <ows:ProviderName>52North</ows:ProviderName>
    <ows:ProviderSite xlink:href="http://www.52north.org/"/>
    ...
  </ows:ServiceProvider>
  <ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get xlink:href="http://geoserver.itc.nl:8080/wps/WebProcessingService"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="DescribeProcess">
      <ows:DCP>
        <ows:HTTP>
```

```
          <ows:Get xlink:href="http://geoserver.itc.nl:8080/wps/WebProcessingService"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="Execute">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get xlink:href="http://geoserver.itc.nl:8080/wps/WebProcessingService"/>
          <ows:Post xlink:href="http://geoserver.itc.nl:8080/wps/WebProcessingService"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
  </ows:OperationsMetadata>
  <wps:ProcessOfferings>
    <wps:Process wps:processVersion="2">
      <ows:Identifier>DouglasPeuckerAlgorithm</ows:Identifier>
      <ows:Title>douglasPeucker algorithm</ows:Title>
    </wps:Process>
    ...
  </wps:ProcessOfferings>
  ...
</wps:Capabilities>
```

DescribeProcess provides this metadata based on the identifier of the designated geoprocess model (e.g. `DouglasPeuckerAlgorithm`). The DescribeProcess request (Listing 3) queries the WPS instance for further metadata on the specific geoprocess model such as input and output parameters. This information is important to trigger the specific geoprocess model appropriately. In the given example (Listing 4), the Douglas-Peucker algorithm requires complex data for the geometries to be processed and literal data (of type double) to indicate the tolerance value the algorithm has to apply to the data.

Listing 3: Example DescribeProcess request retrieving metadata about Douglas-Peucker algorithm.

```
http://geoserver.itc.nl:8080/wps/WebProcessingService?REQUEST=DescribeProcess&Service=WPS&Identifier=DouglasPeuckerAlgorithm
```

Listing 4: Example DescribeProcess response describing the interface for the Douglas-Peucker algorithm.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns:ProcessDescriptions xmlns:ns="http://www.opengis.net/wps/1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsDescribeProcess_response.xsd" xml:lang="en-US" service="WPS"
version="1.0.0"><ProcessDescription xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:xlink="http://www.w3.org/1999/xlink" wps:processVersion="2"
storeSupported="true" statusSupported="false">
    <ows:Identifier>DouglasPeuckerAlgorithm</ows:Identifier>
    <ows:Title>douglasPeucker algorithm</ows:Title>
    <ows:Abstract>Uses JTS implementation. Does not support topological awareness</ows:Abstract>
    <ows:Metadata xlink:title="douglas peucker"/>
    <DataInputs>
      <Input minOccurs="1" maxOccurs="1">
        <ows:Identifier>FEATURES</ows:Identifier>
        <ows:Title>input features</ows:Title>
        <ows:Abstract>Just features</ows:Abstract>
        <ComplexData>
        <Default>
          <Format>
            <MimeType>text/XML</MimeType>
            <Schema>http://schemas.opengis.net/gml/2.1.2/feature.xsd</Schema>
          </Format>
        </Default>
        </ComplexData>
```

```
      </Input>
      <Input minOccurs="1" maxOccurs="1">
        <ows:Identifier>TOLERANCE</ows:Identifier>
        <ows:Title>Tolerance Value for DP Alg</ows:Title>
        <ows:Abstract/>
        <LiteralData>
        <ows:DataType ows:reference="xs:double"/>
        ...
        </LiteralData>
      </Input>
    </DataInputs>
    <ProcessOutputs>
      <Output>
        <ows:Identifier>SIMPLIFIED_FEATURES</ows:Identifier>
        <ows:Title>smooth geometries</ows:Title>
        <ows:Abstract>GML stream describing the smooth feature.</ows:Abstract>
        <ComplexOutput>
        <Default>
        <Format>
          <MimeType>text/XML</MimeType>
          <Schema>http://schemas.opengis.net/gml/2.1.2/feature.xsd</Schema>
        </Format>
        </Default>
        </ComplexOutput>
      </Output>
    </ProcessOutputs>
  </ProcessDescription>
</ns:ProcessDescriptions>
```

Based on these metadata, the client knows where (entry points in service metadata, Listing 2) and how (process metadata, Listing 4) to trigger the designated geoprocess model. The client performs the Execute request (Listing 5) with the designated parameters (geometries and tolerance value). The complex data (i.e. the geometries to be simplified) are included in the request as a reference to a WFS instance. The WPS has to retrieve the data from this location and process them accordingly. This is beneficial to limit the communication overhead between client and WPS instance, as the WPS instance can retrieve the data directly. Additionally, this enables the WPS instance to apply caching strategies, as the service can decide if and when to retrieve the data. Finally, the WPS instance returns the simplified geometries (Listing 6).

Listing 5: Example Execute request for Douglas-Peucker algorithm.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<wps:Execute service="WPS" version="1.0.0" xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
  http://geoserver.itc.nl:8080/wps/schemas/wps/1.0.0/wpsExecute_request.xsd">
  <ows:Identifier>DouglasPeuckerAlgorithm</ows:Identifier>
  <wps:DataInputs>
    <wps:Input>
      <ows:Identifier>FEATURES</ows:Identifier>
      <wps:Reference schema="http://schemas.opengis.net/gml/2.1.2/feature.xsd"
xlink:href="http://geoserver.itc.nl:8080/geoserver/wfs?REQUEST=GetFeature&amp;typename=topp:states&amp;BB
OX=-75.102613,40.212597,-72.361859,41.512517">
      </wps:Reference>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>TOLERANCE</ows:Identifier>
      <wps:Data>
        <wps:LiteralData>2</wps:LiteralData>
      </wps:Data>
    </wps:Input>
  </wps:DataInputs>
```

```
  <wps:ResponseForm>
  <wps:ResponseDocument storeExecuteResponse="false">
    <wps:Output asReference="false">
      <ows:Identifier>SIMPLIFIED_FEATURES</ows:Identifier>
    </wps:Output>
  </wps:ResponseDocument>
  </wps:ResponseForm>
</wps:Execute>
```

Listing 6: Example Execute response for Douglas-Peucker algorithm including process information and simplified geometries.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns:ExecuteResponse xmlns:ns="http://www.opengis.net/wps/1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://geoserver.itc.nl:8080/wps/schemas/wps/1.0.0/wpsExecute_response.xsd"
serviceInstance="http://localhost:8080/wps/WebProcessingService?SERVICE=GetCapabilities&amp;amp;SERVICE=WPS
" xml:lang="en-US" service="WPS" version="1.0.0">
  <ns:Process ns:processVersion="2">
    <ns1:Identifier
xmlns:ns1="http://www.opengis.net/ows/1.1">org.n52.wps.server.algorithm.simplify.DouglasPeuckerAlgorithm</ns1:I
dentifier>
    <ows:Title xmlns:wps="http://www.opengis.net/wps/1.0.0" xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink">douglasPeucker algorithm</ows:Title>
  </ns:Process>
  <ns:Status creationTime="2009-11-16T17:24:14.809+01:00">
    <ns:ProcessSucceeded>The service succesfully processed the request.</ns:ProcessSucceeded>
  </ns:Status>
  <ns:ProcessOutputs>
    <ns:Output>
      <ns1:Identifier xmlns:ns1="http://www.opengis.net/ows/1.1">SIMPLIFIED_FEATURES</ns1:Identifier>
      <ows:Title xmlns:wps="http://www.opengis.net/wps/1.0.0" xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink">smooth geometries</ows:Title>
      <ns:Data>
        <ns:ComplexData schema="http://schemas.opengis.net/gml/2.1.2/feature.xsd" mimeType="text/XML">
          <wfs:FeatureCollection xmlns="http://www.opengis.net/wfs" xmlns:gml="http://www.opengis.net/gml"
xmlns:states="http://www.openplans.org/topp" xmlns:wfs="http://www.opengis.net/wfs"
xsi:schemaLocation="http://www.openplans.org/topp
http://geoserver.itc.nl:8080/geoserver/wfs/DescribeFeatureType?typeName=topp:states http://www.opengis.net/wfs
http://geoserver.itc.nl:8080/geoserver/schemas/wfs/1.0.0/WFS-basic.xsd">
            <gml:boundedBy>
              <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                <gml:coordinates cs="," decimal="." ts=" ">-80.5208,39.7195 -73.3451,45.0061</gml:coordinates>
              </gml:Box>
            </gml:boundedBy>
            <gml:featureMember>
              <states:states fid="states.39">
                <states:the_geom>
                  <gml:MultiPolygon srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                    <gml:polygonMember>
                      <gml:Polygon>
                        <gml:outerBoundaryIs>
                          <gml:LinearRing>
                            <gml:coordinates cs="," decimal="." ts=" ">-79.7635,42.2673 -73.3451,45.0061 -
74.0066,40.7039 -79.7635,42.2673</gml:coordinates>
                          </gml:LinearRing>
                        </gml:outerBoundaryIs>
                      </gml:Polygon>
                    </gml:polygonMember>
                  </gml:MultiPolygon>
                </states:the_geom>
                ...
            </gml:featureMember>
```

```
            </wfs:FeatureCollection>
          </ns:ComplexData>
        </ns:Data>
      </ns:Output>
    </ns:ProcessOutputs>
</ns:ExecuteResponse>
```

This basic sequence of actions can be extended by requesting asynchronous processing or storing of process results on the server side.

The depicted examples of request and response documents are based on 52°North WPS framework (Section 3.1). Nevertheless, the presented workflow should perform successfully on any OGC-compliant implementation of WPS 1.0.0 specification that offers the exercised processes in such a manner.

2.1.2 Semantic Interoperability of WPS

Semantic interoperability is a key requirement to perform geoprocess models meaningfully on the web. The semantics of a geoprocess model such as provided by WPS are unknown, due to the loosely-coupled nature of the Web and the distributed architecture. The links between the Web Services are established during runtime, based on the descriptions of the Web Services. WPS functionality is described through GetCapabilities and DescribeProcess response documents in particular. The syntax is described in the DescribeProcess documents (exemplified Listing 4), but the semantics of the input and output parameters are missing. Geoprocess models offered by WPS instances can only be described on an abstract level. To overcome this problem, one solution is to predefine specific geoprocess models, which are mandatory for any WPS instance. This solution has specific obstacles, as it is hard to determine a set of fixed geoprocess models, which should be offered by WPS instances.

Another solution is *WPS profiles*. These WPS profiles allow the client to identify syntactically and semantically equal geoprocess models provided by WPS instances. WPS profiles are referenced by process descriptions and describe the input and output parameters of a geoprocess model. The WPS profiles provide a common definition of geoprocess models and can be referenced by other WPS instances, which provide a similar geoprocess or a similar set of geoprocesses (syntactically and semantically). To give an example, let us assume there are two different implementations of a buffer algorithm and both published as WPS-based geoprocess models sharing the same interface (i.e. same input and output parameters). As both geoprocess models refer to the same WPS profile, they become interoperable (i.e. sharing the same interface) but also their functionality becomes comparable to the client. The client can select the appropriate WPS-based geoprocess model based on the quality of the process output and the performance of the geoprocess model. However, matching the semantics of the offered geoprocess model with the WPS profile is the responsibility of the service provider. From a technical perspective, WPS profiles are descriptions of WPS-based geoprocess model (i.e. defining input and output parameters), which are web-accessible and are identified by an OGC Uniform Resource Name (URN).

Nash (2008) specified an initial set of these WPS profiles describing the most common GIS operations. Ostlaender (2009) also used WPS profiles to describe WPS-based geoprocess models in an SDI for spatial decision support.

WPS profiles are still a subject to research, as it is hard to determine if two geoprocess models are equal and if they can be matched to the same WPS profile. A geoprocess might appear equal for instance to the service provider, but not to the user. Nash (2008) and Bucher & Jolivet (2008) conclude that for meaningful WPS profiles the geoprocessing functionality has to be classified by a commonly agreed taxonomy of geoprocess models which is currently lacking. Finally, an agreed classification of

functionality formalized as ontologies is required to enable semantic interoperability and semantic reasoning. Ontologies of such geoprocesses can be built in a coarse grain (Lemmens, 2006) or a fine grain (Lutz, 2007) manner.

2.2 Geoprocessing Workflows

The complexity of geodata and the complexity of the problem domain often require geoprocess models consisting of multiple steps. These complex geoprocess models can be implemented by chaining multiple WPS instances to create a value-added geoprocessing workflow.

In particular, we aim at enabling the full potential of OGC Web Services as an integration platform. This will be achieved when applications and business processes can be composed to perform complex interactions using a standardized process integration approach. Therefore, Geoprocessing Workflows regarded as service chains are one of the key concepts in enabling value-added chains in SDIs (Alameh, 2003).

A *workflow* is defined by ISO as an "automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" (ISO/TC 211, 2005). For this chapter, the workflow participants are understood as Web Services, which pass information from one workflow participant (i.e. Web Service) to another especially across enterprise boundaries.

A *geoprocessing workflow* combines the concept of geoprocessing (defined in Section 2) and the concept of workflows. This chapter defines geoprocessing workflow as an automation of a geoprocess model, in whole or part, during which information is passed from one Geoprocessing Service to another according to a set of procedural rules using standardized interfaces.

In other words, geoprocessing workflows integrate data and services in an interoperable way, where each part of the workflow is responsible for only a specific task, without being aware of the general purpose of the workflow. Due to the distributed nature of geographic data and by using the presented definition, geoprocessing workflows provide a flexible means of processing highly distributed and complex data for a wide variety of uses across enterprise borders.

Section 2.2.1 will describe how geoprocessing workflows are modeled as service chains in SOAs and will introduce the different patterns of chaining. Moreover, the technologies for realizing such service chains are introduced (Section 2.2.2).

2.2.1 Realizing Geoprocessing Workflows as Service Chains

Workflows implemented as service chains are one of the key concepts in enabling value-added chains in SDIs (Alameh, 2003). The ISO19119/Service Architecture standard defines service chaining as: "A sequence of services where, for each adjacent pair of services, occurrence of the first action is necessary for the occurrence of the second action" (ISO/TC 211, 2005).

A service chain is a directed graph, since the input of one service depends on the output of another service. A directed graph is defined as a set K of ordered nodes and a set E of edges, where each edge $e(u,v) \in E$ has a direction and consists of a node pair (u,v) where $u,v \in K$.

The nodes in a directed graph represent service entities and the arcs represent the service interactions. Directed acyclic graphs (DAG) are special types of directed graphs. The definition of a directed graph

from above has to be extended with the constraint that for any node *t*, there is always an empty directed path that starts and ends on *t*.

However, some service chains require iterations and for this reason the graph has to be cyclic and therefore has to make use of conditions in the control function to address convergence.

In addition, there are four more characteristics of a service chain according to ISO19119 (ISO/TC 211, 2005):

- Parallel or serial sequences
- Variations in the links between nodes reflecting different methods of transporting data or invoking the service
- Parameters in nodes
- Pull processing vs. push processing.

Besides, there are three different architectural patterns for service chains defined as a foundation for geoprocessing workflows according to Alameh (2003) and ISO19119 (ISO/TC 211, 2005):

- Transparent chaining
- Translucent chaining
- Opaque chaining.

In the transparent chaining pattern, the knowledgeable user defines a service chain by specifying the different participants of the service chain and by defining the specific sequence of interaction. In particular, the user is responsible for discovering and evaluating available services as well as for defining the execution order, invoking the services and pass around process results as inputs. Furthermore, the user has to make sure that input and output messages have to be compatible and all required resources are available. Since all service details are visible to the user, this pattern is called transparent chaining. Figure 2 presents this pattern as an UML collaboration diagram.

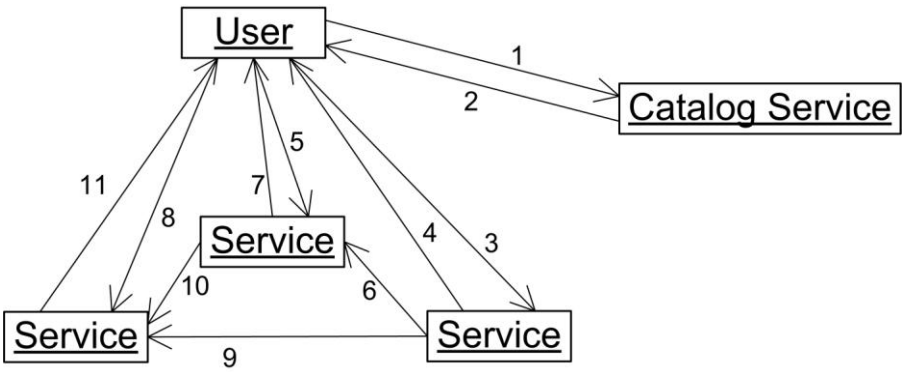


Figure 2: Transparent chaining pattern.

In Step 1, the user sends a search request to a catalog service in order to discover service availability. The catalog service returns metadata about services fulfilling the search request. The user creates an execution order and invokes the first service in Step 3. The processed results (or a reference) are returned back to the user in Step 4. These results (or a reference) are passed in the second service in Step 5. If the user supplies a reference, the service has to obtain the actual data in Step 6 from the previous service. Again, the processed results (or a reference) are returned back to the user as can be seen in Step 7. In Step 8, the user invokes the third service with the results from the two previous services. If a reference to actual data

is delivered, the third service requests the actual data from the corresponding in Step 9 and Step 10. After processing, the final results are returned to the user in Step 11.

The translucent chaining pattern allows a user to execute a predefined service chain managed by a workflow service. In this pattern, the service chain is already abstractly predefined and stored on a workflow engine. The user is aware of all participants of the service chain, but does not have to deal with the execution order or with passing around processing results. But since the user knows all participating services, he is able to poll the current status of each participating service (if supported by the service). Figure 3 gives an overview of this pattern.

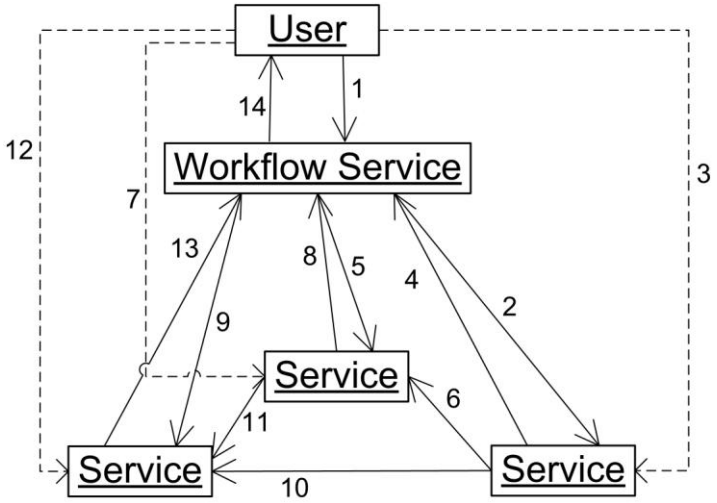


Figure 3: Translucent chaining pattern.

In Step 1, the user invokes an existing chain on the workflow service. The workflow service starts now the predefined execution order. The first service is invoked in Step 2. Since the user knows, that this service is invoked first, he/she can poll the current processing status of the service (Step 3). The processed results (or a reference) are returned back to the workflow service in Step 4. These results (or a reference) are passed in the second service in Step 5. If the workflow service supplies a reference, the service has to obtain the actual data in Step 6 from the previous service. Again, the user can poll the status in Step 7 and after processing, the results (or a reference) are returned back to the workflow service as can be seen in Step 8. In Step 9, the workflow service invokes the third service with the results from the two previous services. If a reference to actual data is delivered, the third service requests the actual data from the corresponding in Step 10 and Step 11. As seen before, the user is enabled to poll the current service status in Step 12. After processing, the final results are returned to the workflow service in Step 13 and from there back to the user in Step 14.

The opaque chaining pattern exposes a service chain and thereby a geoprocessing workflow as a single service and hides all details from the user. The user is not even aware of the fact that the aggregate service hides a chain nor is the user aware of the types of services being used. Therefore, the aggregate service is responsible for all service coordination. Figure 4 describes this pattern.

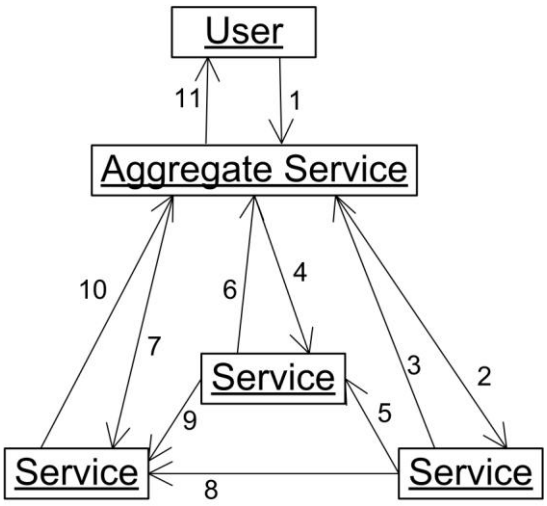


Figure 4: The opaque chaining pattern.

In Step 1, the user invokes an aggregate service unaware of the implementation details (e.g. if the aggregate service uses a service chain and what kind of services are involved). The aggregate service starts now the predefined execution order at this point, so the first service is invoked in Step 2. The processed results (or a reference) are returned back to the aggregate engine in Step 3. These results (or a reference) are passed in the second service in Step 4. If the aggregate service supplies a reference, the service has to obtain the actual data in Step 5 from the previous service. Again, after processing, the results (or a reference) are returned back to the aggregate service as can be seen in Step 6. In Step 7, the aggregate service invokes the third service with the results from the two previous services. In case a reference to actual data is delivered, the third service requests the actual data from the corresponding service as presented in Step 8 and Step 9. After processing, the final results are returned to the aggregate service in Step 10 and from there back to the user in Step 11.

2.2.2 Technology for Service Chaining

As defined above, a geoprocessing workflow can be seen as a geoprocess model. Typically a geoprocess model incorporates different functionality. If such a geoprocess model is implemented as a SOA, this functionality is provided by different Web Services. Thus to realize such geoprocess models as geoprocessing workflows, the principles of a SOA play an important role.

Web Service Composition and *Web Service Orchestration* describe workflows from a SOA perspective and are therefore relevant concepts for geoprocessing workflows. According to Alonso et al. (2004) Web Service Composition combines several Web Services to one workflow to represent a desired business process. Alonso et al. (2004) define a composite service following the opaque chaining pattern and entitle the act of combining Web Services as *Web Service Composition*. The interaction of the Web Service in the workflow can be handled by *Web Service Orchestration*, which is a description of how combined Web Services interact on the message level (Peltz, 2003). This description includes the business-logic and execution order. *Web Service Orchestration* is often realized by a central orchestration engine coordinating the Web Service interaction according to a predefined workflow description and thereby falls into the opaque architecture pattern. In particular, this approach allows the Web Services to be held loosely coupled (Weerawarana, Curbera, Leymann, Storey, & Ferguson, 2005).

Several orchestration languages are available (Staab et al., 2003). XPDL as a Workflow Management Coalition workflow language, YAWL as a popular open source language (van der Aalst and Hofstede,

2005) and the Business Process Execution Language (BPEL) (Andrews et al., 2003) are the most relevant. In particular BPEL has advanced to a de-facto standard in the mainstream IT-world (van der Aalst et al., 2005). It is an OASIS standard for describing workflows based on elementary tasks implemented as Web Services using an XML-encoding. In particular, these so-called BPEL scripts describe the roles involved in the message exchange, supported port types and orchestration information of a process. Therefore, BPEL seems to be an appropriate candidate to create Geoprocessing Workflows in SOAs, as it enables the composition of workflows based on loosely coupled services.

2.2.3 Current Research & Future Challenges in Geoprocessing Workflows

Designing and realizing geoprocessing workflows are still a subject to current research and also remain future challenges. Leaving the semantic aspect out of scope (as for example addressed by Cardoso & Sheth (2003) and from a geo-related perspective addressed in Roman & Klien (2007)), some research regarding geoprocessing workflows has been accomplished recently. Kiehle, Greve, & Heier (2006) describe a rule-based approach and theoretical foundations for the orchestration of Geoprocessing Services. Granell, Gould, & Ramos (2005) as well as Friis-Christensen et al. (2006) provided conceptual foundations, such as elementary workflow patterns for Geospatial Web Services. Friis-Christensen, Lucchi, Lutz, & Ostlaender (2009) introduced the term *Distributed Geographic Information Processing* (DGIP) and presented different architectural patterns for chaining of Geoprocessing Services. Besides, Weiser & Zipf (2007) focused on the application side by utilizing BPEL. Schaeffer (2008) showed an application of the opaque chaining pattern to geoprocessing workflows, by extending the WPS with a transactional interface (WPS-T). WPS-T provides a mechanism to deploy geoprocessing workflows, which are modeled as a service chain, as single WPS-based geoprocess models at runtime. Consequently, geoprocessing workflows are encapsulated as WPS-based geoprocess models and can be triggered using OGC-compliant messaging. The approach is based on BPEL but extensible through profiling the deployment procedure for different workflow languages.

On the standardization level, the OGC Interoperability and Specification Programs have also produced a significant body of knowledge and experience in designing, implementing and deploying Web Services (Keens, 2007; B. Schaeffer, 2009; Werling, 2008).

Besides the current research, still some challenges remain open, which can be categorized in three types. At first, performance is an issue in geoprocessing workflows. The orchestration of services and distributing data across a SOA comes always with an overhead. Suitable workflow patterns and efficient data encodings have to be found. Use of reference data might be one solution for centralized workflow approaches. Friis-Christensen, Lucchi, Lutz, & Ostlaender (2009) investigated several patterns, which are promising to be elaborated in the future such as efficient encoding of data for geoprocessing workflows.

At second, to improve the sustainability and the commercial potential of SDIs, security and licensing of specific geoprocessing workflows might become an issue as also foreseen by INSPIRE (2007). Consequently, to chain secured and licensed service, challenges regarding the delegation of rights between user and workflow engines and workflow partners arise. One solution to solve this issue in an interoperable way using open standards has been presented by Schaeffer (2009) for opaque workflows, but has to be extended to other workflow patterns.

Finally, the semantic annotation of Geoprocessing Services and thereby improving the semantic interoperability is also one major challenge for the future. Therefore, semantic annotations for geoprocessing workflows are promising to enable the automated creation of geoprocessing workflows.

2.3 Quality of Service in Distributed Geoprocessing

In legally mandated SDIs such as the Infrastructure for Spatial Information in the European Community (INSPIRE) (legally binding since 2007 (INSPIRE, 2007a)) several performance requirements are defined for the so-called *network services* (INSPIRE, 2008, 2007b). For example, search queries need to be answered within 3 seconds and services must be able to handle up to 30 of these queries at the same time. To meet such performance requirements in peak times, scalable solutions have to be found which have not been implemented in SDIs yet. Furthermore, monitoring the performance of loosely coupled Web Services in a distributed architecture and the ability to react quickly on service quality fluctuations is overall an essential skill for service consumers and service providers in future business models. However, existing OGC specifications and standards do not support any kind of service quality enforcement functionality on a specification level.

Quality of Service (QoS) is defined by the International Telecommunications Union (ITU-T) as ”the collective effect of service performances, which determine the degree of satisfaction of a user of the service” (ITU-T, 2001) and it ”is characterized by the combined aspects of service support performance, service operability performance, service integrity and other factors specific to each service” (ITU-T, 1994). Performing geoprocess models over the web raises serious QoS issues that did not occur in case of desktop-based geoprocessing. Firstly, the service quality is strongly affected by general service and network performance characteristics. Secondly, the overall performance of a geoprocess model is implementation and algorithm specific and depends on the amount of involved data as well as the complexity of the input data and the available computational resources.

Following the argumentation of Ran (2003), J. Lee, Jeon, W. Lee, Jeong, & Park (2003) and Geraci (1991), the reliability and performance of a Web Service can be identified as the two major issues of QoS in the context of a SOA. Additionally, several other quantifiable service properties can be identified like transaction-, security-, costs- and functionality-related QoS. The remainder of this section will describe the major QoS issues in more detail.

2.3.1 Ensuring Reliability

Reliability is the ability of a service to process service requests and to perform its required functions under stated conditions for a specified time interval. The reliability can be quantified in terms of availability, accessibility, capacity, scalability, robustness, flexibility and accuracy of a Web Service.

Among other things, common Server Load Balancing (SLB) and failover mechanisms can be utilized to increase the different aspects of reliability (such as availability and capacity of a Web Service). For example in so-called high-availability clusters redundant or standby computers are used for the purpose of providing high availability of Web Services. All these mechanisms and technologies are common in the mainstream IT world to improve reliability-related QoS and could be applied for Geoprocessing Services as well.

However, especially when processing large amounts of data or performing large-scale simulations the highest impact on the degree of satisfaction of the user of the Geoprocessing Service is caused by the duration of the geoprocess model itself.

2.3.2 Ensuring Performance

The performance describes the speed at which service requests are completed. Performance can be measured in combination of throughput, latency, response-, execution- and transaction-time. Several

reliability and performance related parameters are strongly related and define the same quality aspect of a Web Service but with other measuring units.

Performance is reported to be crucial in the context of Geoprocessing Services (Baranski, 2008; Bernard, Craglia, Gould, & Kuhn, 2005; Di, A. Chen, Yang, & Zhao, 2003; Kiehle, Heier, & Greve, 2007; Scholten, Klamma, & Kiehle, 2006; Tu et al., 2004). The performance of a Geoprocessing Service heavily depends on the availability of sufficient computational resources, that are primary responsible for the speed in completing a single geoprocess in a convenient period of time. The resource requirements of computational jobs for submission to Grid Computing infrastructures are described in Anjomashoaa et al. (2005). The defined resource requirements and therefore the corresponding performance-related QoS parameters also apply to Geoprocessing Services, such as the provision of three CPUs and 20 GB storage in a specific time frame. As mentioned in Section 2.3, beside the availability of computational resources the overall performance of a geoprocess model is also implementation and algorithm specific and depends on the amount of involved data as well as the complexity of the input data.

In the context of Geoprocessing Services some research has been carried out to address the performance of large-scale computations by Grid Computing and Cloud Computing. Grid Computing overlaps with some concepts of Cloud Computing (Hartig, 2009). Both, Grid and Cloud Computing infrastructures provide a distributed network infrastructure for scaling applications by sufficient storage and computational capabilities. However, Grid Computing is typically applied by the scientific community for large-scale computations (e.g. a global climate change model or the aerodynamic design of engine components). Whereas Cloud Computing enables small and medium-sized companies to deploy their web-based applications in an instant scalable fashion without the need to invest in large computational infrastructures for storing large amounts of data and/or performing complex processes (Myerson, 2008). As a consequence, national and international Grid Computing infrastructures (for example the Worldwide LHC Computing Grid) are typically funded by the government and operated by international joint research projects, whereas cloud infrastructures are operated by large-sized enterprises under economic aspects, such as Amazon or Google, enabling smaller companies to use their infrastructure.

Baranski (2008) and Lanig, Schilling, Stollberg, & Zipf (2008) extended the work of Di et al. (2003) and accomplished first experiments using Grid Computing technology for improving computational performance by distribution and parallel execution of processes. In the OGC Web Services, Phase 6 (OWS-6) initiative of the OGC Interoperability Program (Baranski, 2009), a WPS profile for accessing applications in Grid Computing infrastructures has been reviewed. Baranski, Schaeffer, & Redweik (2009) presented and tested an approach of bringing the OGC Web Processing Service to a Cloud Computing infrastructure. The SLA4D-GRID research project[2] is designing and realizing a Service Level Agreement (SLA) layer for the Germany's national Grid infrastructure D-Grid. SLAs are negotiated business contracts between service consumers and providers defining service qualities (beside other contractual elements like costs and penalties) that will be guaranteed by a service provider. The integration of SLA in mature grid middleware and the application of advanced resource reservation mechanisms in Grid Computing infrastructures will enable providers to guarantee promised service qualities. Providing sufficient computational resources for Geoprocessing Services (namely the WPS) is one important use case within the SLA4D-Grid project.

2.4 Legacy System Integration

Legacy systems encapsulate existing functionality, which performs well and is tested thoroughly. In particular they incorporate an exhaustive variety of geoprocess models, which is mostly available in desktop-based GIS or web-accessible in a proprietary (i.e. non-standardized) way. However, standardized

[2] SLA4D-GRID project website: http://www.sla4d-grid.de/.

Geoprocessing Services (e.g. OGC WPS implementations) still lack manifold functionality. To prevent a reimplementation and enable further usage of existing functionality in a sustainable way, an approach from mainstream SOA called *wrapping* (Papazoglou & Heuvel, 2007; Erl, 2005) can be used for Geoprocessing Services as well. A wrapper translates incoming requests to commands in such a way, that the wrapped legacy system understands them and performs accordingly. It is important to note that the standardized Web Service interface remains unchanged and therefore interoperability is ensured.

In the following the requirements for such wrapping will be analyzed and an architecture will be developed to extend a WPS implementation for wrapping the GIS functionality of a legacy system. In particular, creating descriptions for WPS-based geoprocess models automatically for functionality of a legacy system will be discussed. An implementation based on the 52°North WPS framework (as a wrapper) and GRASS GIS (as a legacy system) will be described. Besides, disadvantages will be pointed out and solutions to tackle theses problems will be proposed.

2.4.1 Wrapping Requirements

To successfully wrap existing functionality as WPS-based geoprocess models, several requirements have to be met by the legacy system. One essential and mandatory requirement for the legacy system is the ability to invoke its functionality unsupervised for two reasons. At first, machine-to-machine communication and automatic service invocation are required by SOA and its Web Service environment (Erl, 2005). This is not ensured if human interaction is required to invoke specific functionality. At second, the stateless communication pattern of a SOA (Erl, 2005) requires that all parameters necessary for successful processing are sent by the time of invocation. All further interactive communication would break the stateless communication.

This capability is surprisingly frequent for legacy GIS systems, although originally mostly intended for batch processing (e.g. do the same raster classify for the satellite images of a whole year without entering the same input parameters for each single image). It is implemented differently among legacy GIS ranging from simple scripting interfaces like Linux shell or DOS batch programming to high-level programming APIs like Python or JAVA.

Another requirement is that the functionality of a legacy system has to be described by a WPS ProcessDescription document (Section 2.1). Creating process descriptions manually can become a tedious task, especially regarding the variety of functionalities offered by a Desktop GIS. For example, GRASS combines approximately 400 modules, which each can be offered as a geoprocess model (Neteler & Mitasova, 2008). If the legacy system offers structured and therefore machine-readable process descriptions, a semi-automatic transformation into the XML-based WPS DescribeProcess documents is generally possible (see details for this transformation later on). The wrapping approach is applicable nevertheless, if the legacy system does not provide structured process descriptions.

To improve the performance of the wrapping architecture (i.e. accessing a WPS-based geoprocess model based on functionality provided by a legacy system), the legacy system and the WPS may perform processes concurrently. Running processes concurrently requires a fine granular modularization with independent multiple executable modules. This improves the performance in contrast to legacy systems which can only be instantiated once at the same time (such as desktop-based GIS). Nevertheless, the execution of parallel incoming requests can be prevented by refusing requests (e.g. 'server busy') or by sequentially processing them.

2.4.2 Creating Process Descriptions for Legacy System Integration

To supply the wrapped functionality of the legacy system with applicable data, process descriptions have to be created. These process descriptions specify the syntax of the specific geoprocess model and allow the client to configure the geoprocess model accordingly. As mentioned above an automated approach for creating process descriptions might be feasible if the legacy system already offers structured information (i.e. encoded in XML format). In this case an Extensible Stylesheet Language Transformation (XSLT) document can be designed to transform the legacy descriptions into DescribeProcess documents. Creating such an XSLT filter might appear as a huge effort to publish a single functionality of the legacy system, but considering wrapping the functionality of a complete legacy system (incorporating 100 processes or more) such a XSLT filter might be the applicable solution.

This XSLT-based approach does not work out-of-the-box. Several aspects have to be considered carefully. First, not all functionality offered by a legacy system is applicable as a WPS-based geoprocess model. For instance, desktop-based functionality for visualization is not suitable as a WPS-based geoprocess model and is considered to be a client task. Assuming a structured description of the functionality offered by the legacy system is available (e.g. encoded in XML), the XSLT filter is not able to identify which functionality is applicable to be published as WPS-based geoprocess model. The semantics of the specific functionality of the legacy system is not available. Consequently, inappropriate functionality has to be excluded manually, before the DescribeProcess documents are generated.

Second, the WPS distinguishes between simple and complex input parameters, whereas most legacy systems have a single type of input parameter. Complex input parameters for WPS incorporate geodata which are encoded in a data format, using a specific mime type and in case of XML a specific schema. Input parameters for legacy systems are usually described by a single String. No further information is necessary for processing as it can be assumed that it is already imported into the workspace in a legacy system internal data format. The name is sufficient to identify the input dataset and therefore it is ready to use without taking care of further data format related issues. Again, the XSLT filter is not capable of identifying which parameter is complex and needs to be described in greater detail. Listing 7 and Listing 8 show the different complexities between legacy system functionality (in this case GRASS) and WPS DescribeProcess documents.

Finally, it can be concluded that a fully automated approach for creating process descriptions is not feasible at the moment. The required semantics cannot be extracted from the description of the legacy system and interpreted by the XSLT filter. Nevertheless, a semi-automated approach using a manual selection of applicable geoprocess models and performing a XSLT filter on the specific process description is less time-consuming than creating all descriptions manual from scratch.

2.4.3 Wrapping Architecture

This section describes the conceptual architecture incorporating functionality of a legacy system in a WPS. To successfully wrap such functionality, one aspect has to be considered, which is specific to distributed architectures. As stated before, the communication between Web Services is stateless (Erl, 2005), whereas legacy systems are able to maintain a certain state during several processing tasks. In case of legacy systems it is possible to store intermediate and final results inside the legacy system using a workspace-based concept incorporating also for instance information about the applicable Coordinate Reference Systems (CRS).

Hence, for accessing functionality of a legacy system inside a WPS, a workspace has to be created for each WPS Execute request. A workspace encompasses a CRS definition for the data to be imported and the data itself. Importing data into the workspace normally includes a transformation from the provided

common format (such as GML or shape file format) into the internal data structure required by the legacy system. For providing the user with a result (mostly in a common data format again) the data has to be transformed again and exported to the requested data format. Additionally, if this response represents an intermediate result which has to be further processed on the same WPS, it nevertheless has to be sent back to the client and then sent back to the server and imported again. Creating workspaces and importing and exporting data is always required as already mentioned. This creates an overhead on the architecture regarding data transfer and processing time, which cannot be avoided without harming the interoperability (e.g. by introducing additional parameters, which keep track of processing sessions). Some legacy systems require no data import for certain formats and work directly on the available data by referencing it into the workspace (e.g. GRASS for most vector and raster data). Still a workspace has to be created, however this limits the effort of importing, which in most cases is beneficial to improve performance.

Based on the mentioned implication of stateless Web Service communication and the requirements outlined in the previous section the following architecture is proposed (Figure 5).

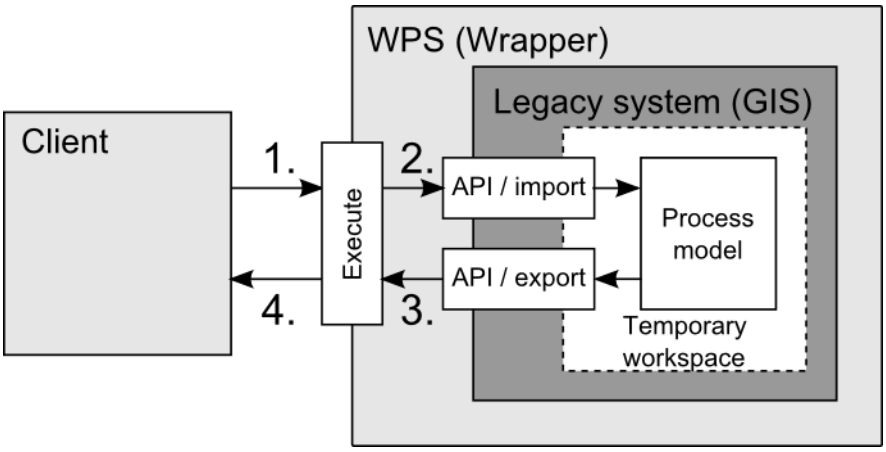


Figure 5: Architecture for wrapping functionality of legacy systems inside WPS.

All necessary parameters and the data to be processed or a reference to the data (e.g. URL) are incorporated in the Execute request which is sent by the client (Step 1). The data can be read directly from the request or retrieved from a reference incorporated in the request. Although it is possible to download such referenced data by functionality of legacy systems (e.g. a WFS connector) it should not be used due to security reasons. This prevents the uncontrolled infiltration of malicious code directly into the hosting environment. The WPS which runs autonomously as a Web Service has no direct access to the hosting environment. Unfortunately, this approach requires an additional import step whereas a WFS connector could import the data directly into the created workspace. After creating and importing the provided data (Step 2), the designated functionality of the legacy system is invoked and the data is processed accordingly. After exporting the result in the requested data format, it is handed back to the WPS (Step 3), which sends it back to the client (Step 4). Alternatively, the WPS directly understands the result's data format and is able to transform it into the requested format (e.g. in the case of SEXTANTE as legacy system).

2.4.4 Implementing the Wrapping Architecture

To demonstrate the approach of wrapping existing functionality as WPS-based geoprocess models, this section illustrates this approach by the example of two legacy systems:

- GRASS (Brauner & Bastian Schaeffer, 2008)

- SEXTANTE (see Diaz et al. 2008 for details).

Both legacy systems are incorporated in the 52°North WPS framework (Section 3.1). Additionally, the 52°North WPS framework is able to provide a subset of ESRI's ArcGIS Server functionality, as demonstrated by Müller, Vogel, & Bernard (2009).

GRASS

GRASS is an Open Source Desktop GIS for vector and raster analysis whose beginnings date back to the 1980ies. An exhaustive overview of GRASS' history is provided in Neteler & Mitasova (2008). Functionality in GRASS is grouped in modules which can be invoked using command line tools or the desktop application of GRASS.

GRASS provides structured process descriptions in XML, which can be created by invoking a GRASS module with the '--interface-description' parameter. These descriptions can be transformed by an XSLT filter operation into WPS DescribeProcess documents (as described in Section GHM). GRASS modules group different functionality. It is therefore easy to exclude specific functionality by restricting the wrapper to specific modules which are not applicable in SDI (like visualization modules which all starts with a 'd.'). As long as GRASS modules require only a single complex input parameter (the geodata) they are labeled as 'input' and the XSLT filter is able to identify them as complex parameter and adds additional data format information. Modules with more than one complex input (like an overlay of two data sets) require manual adoptions. Listing 7 and Listing 8 show excerpts of process descriptions for GRASS and the WPS; the different complexities are highlighted in grey.

Listing 7: Example of a complex input parameter (geodata) in GRASS GIS process descriptions.

```
<parameter name="input" type="string" requires="yes" multiple="no">
    <description>
        Name of input vector map
    </description>
</parameter>
```

Listing 8: Example of a complex input parameter (geodata) in a WPS DescribeProcess document.

```
<Input minOccurs="1" maxOccurs="1">
    <ows:Identifier>input</ows:Identifier>
    <ows:Title>Polygon to be buffered</ows:Title>
    <ows:Abstract>The geometries to buffer</ows:Abstract>
    <ComplexData>
        <Default>
            <Format>
                <MimeType>text/XML</MimeType>
                <Schema>http://schemas.opengis.net/gml/2.1.2/feature.xsd</Schema>
            </Format>
        </Default>
        <Supported>
            <Format>
                <MimeType>text/XML</MimeType>
                <Schema>http://schemas.opengis.net/gml/2.1.2/feature.xsd</Schema>
            </Format>
        </Supported>
    </ComplexData>
</Input>
```

Wrapping of GRASS by the 52°North WPS framework is illustrated in the architecture depicted in Figure 6.

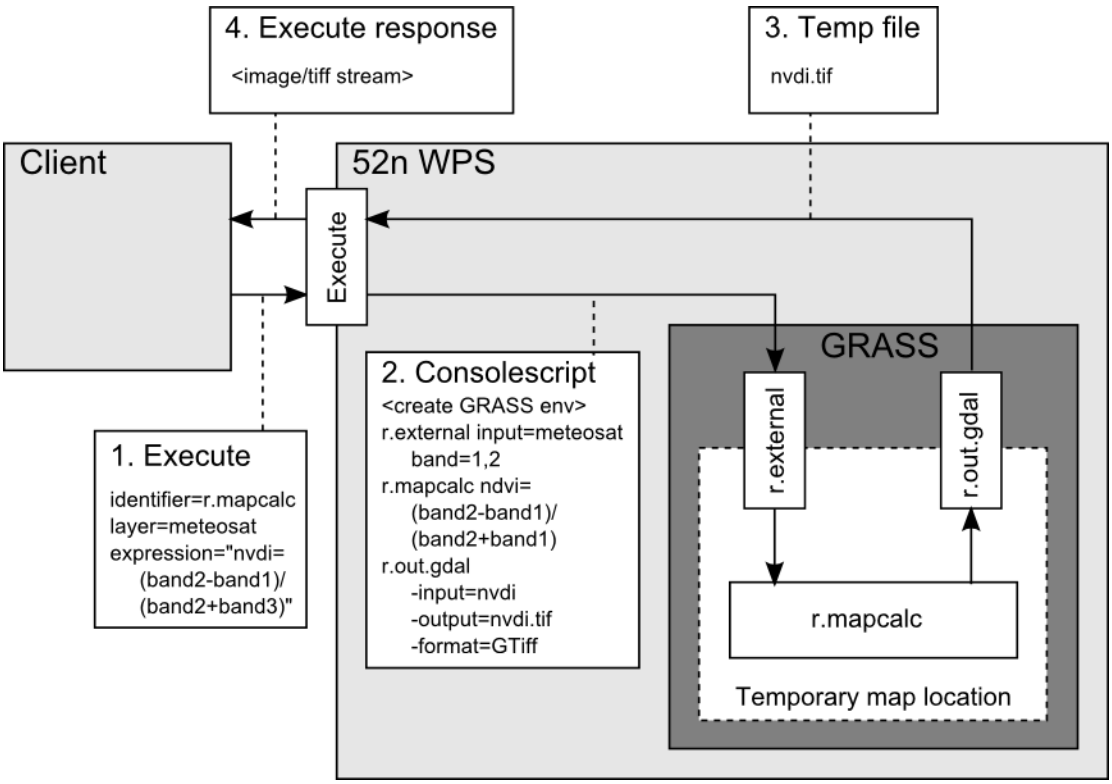


Figure 6: Wrapping the NDVI calculation of GRASS by the 52°North WPS framework.

After receiving the WPS Execute request from a client (Step 1), the WPS downloads the appropriate input data and saves it as a temporary file on the hard disk or as a binary input stream in the main memory depending on the size and the type of data. The WPS then creates a console script (Step 2) with the appropriate GRASS commands to create a workspace, imports or references the complex input data into the workspace and processes the data. Finally, the WPS exports the process result in the requested data format into a temporary file (Step 3) and provides the client with the result (Step 4). GRASS offers modules to request data from external Web Services. Nevertheless, they are not used here due to the security reasons described in the architecture section.

SEXTANTE

SEXTANTE is a JAVA-based library incorporating over 200 different geoprocess models (ranging from simple buffer calculations to complex slope analysis). The integration into the 52°North WPS framework follows the same core architectural principles for wrapping as described above, but is nevertheless easier due to the following reasons. First, for creating of the DescribeProcess documents of the SEXTANTE functionality no XSLT filter is necessary. This is due to the fact, that SEXTANTE and 52°North WPS framework can communicate internally as they are both implemented in JAVA. JAVA allows the 52°North WPS framework to inspect the syntax of the SEXTANTE functionality using reflection. Thereby these DescribeProcess documents can be created at runtime and no manual interaction is required. This is also due to the fact, that SEXTANTE is only designed to provide processing functionality and no visualization for instance. This was one of the reasons, why automated extraction and selection of applicable geoprocess models has been considered to be problematic (Section 2.4.2). Second, by using Java for internal communication, the WPS can directly invoke the SEXTANTE functionality.

Finally, SEXTANTE uses common data formats compatible to the ones used for the WPS. Hence, no workspaces or data transformation are required for successful wrapping.

3 Applications

Distributed geoprocessing is addressed in many projects. It is the backbone to provide information based on distributed geospatial data using geoprocess models. To illustrate the wide range of projects using distributed geoprocessing based on WPS Table 1 provides an overview of relevant projects. Brauner et al. (2009) provide a comprehensive overview about ongoing and finished applications and projects dealing with geoprocess models in an SDI context. Table 1 groups these projects according to their aim.

Table 1: Overview of current geoprocessing projects (Brauner et al. (2009)).

Category	Project Name	Website	Selected literature
Raster-based processing	AWARE	www.aware-project.net	Granell, Diaz, & Gould (2007)
Grid computing	GDI-Grid	www.gdi-grid.de	Baranski; Lanig et al. (2008)
	SEE-GEO	edina.ac.uk/projects/seesaw/seegeo	Koutroumpas and Higgins 2008
	ESDISP		Di et al. (2003)
Automated Generalization	DURP ondergronden	www.durpondergronden.nl	Foerster & Stoter (2006)
	WebGen	webgen.geo.uzh.ch	Foerster et al.; Neun, Burghardt, & Weibel (2008)
Schema Translation in INSPIRE			Lehto (2007)
Wrapping existing (desktop) GIS			(Brauner, 2008; Brauner & Bastian Schaeffer, 2008; Diaz, Costa, Granell, & Gould, 2007)
Spatial statistics	INTAMap	www.intamap.org	Henneboehl & Pebesma; de Jesus, Hiemstra, & Dubois (2008)
Spatial Decision Support Systems (SDSS)	OK-GIS	www.ok-gis.de	Stollberg & Zipf (2007)
	ORCHESTRA	www.eu-orchestra.org	(Friis-Christensen, Ostlander, Lutz, & Bernard, 2007)
	AWARE	www.aware-project.net	Diaz et al. (2007)
	ImmoSDSS_RLP	www.i3mainz.fh-mainz.de/Article300.html	Stollberg & Zipf (2008)
SDSS / Multicriteria Evaluation	SoKNOS	www.soknos.de	Müller et al. (2009)

This section presents different applications, which are applied in some of these projects to realize distributed geoprocessing. The foundation for such projects is an implementation of the WPS interface specification to publish the geoprocess models on the Web. Section 3.1 describes the architecture of such a WPS implementation by the example of the 52°North WPS framework. To access the available geoprocess models client applications are necessary. The geoprocess models can be accessed in different ways using a common desktop-based GIS such as *User-friendly Desktop Internet GIS* (uDig[3]) or mass

[3] uDig website: http://udig.refractions.net.

market applications such as Google Earth. Both applications have a different set of functionality to access and configure these geoprocess models. The WPS client application based on uDig allows expert users to thoroughly configure the specific geoprocess model. Whereas mass market applications such as Google Earth only aim ordinary users requesting specific information without any knowledge about the underlying architecture. Both client applications are described in Section 3.2 and Section 3.3 respectively.

3.1 52°North WPS Framework

This section describes the architecture of the 52°North WPS framework. This framework exemplifies the design challenges of implementing the OGC WPS interface specification (OGC, 2007). Another example of an implementation of the WPS interface specification version 1.0.0 is PyWPS (Cepicky & Becchi, 2007).

The development of 52°North WPS framework has been started in 2006, when the WPS interface specification was released under version 0.4.0. With the advancement of the specification the implementation has matured and is compliant to the most recent version 1.0.0 of the specification. The 52°North WPS framework is available as Open Source under the GNU General Public License at the 52°North Open Source initiative. The implementation is based on Open Source libraries such as GeoTools (Turton, 2008) and XMLBeans (xmlbeans.apache.org). The 52°North WPS framework is designed as a JAVA servlet application, which can be deployed for instance on an Apache Tomcat server.

The design goal of the 52°North WPS framework was to be extensible and pluggable in terms of geoprocess models and data handlers. Service providers are able to incorporate their own geoprocess models and to distribute the results regarding in their own data format using customized data handlers during runtime. The layout of the architecture is depicted in Figure 7.

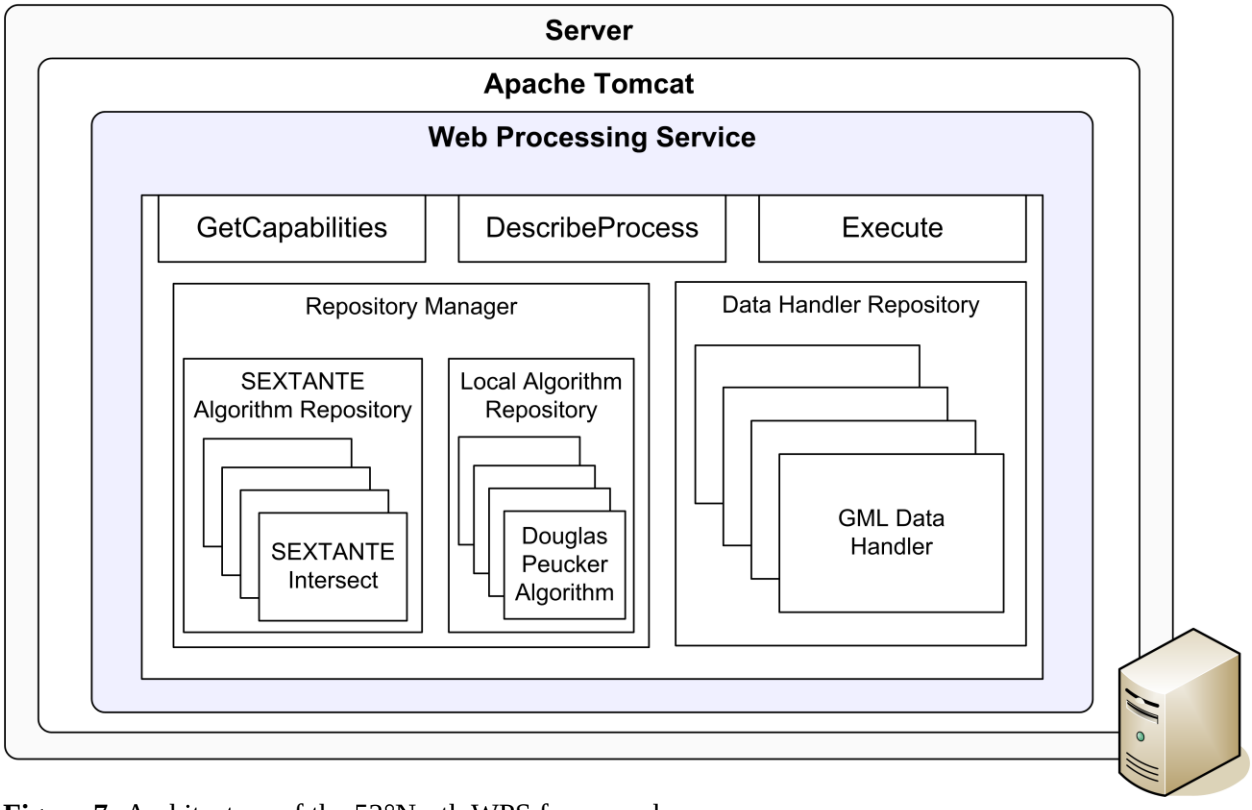


Figure 7: Architecture of the 52°North WPS framework.

To enable this extensible and pluggable architecture, three major entry points are defined:

- Algorithm Repository

- Repository Manager
- Data Handler Repository.

These entry points are initialized based on a configuration file, which can be changed by the service administrator of the WPS using a graphical configuration tool. Whenever the configuration changes, the entry points are re-initialized to realize the pluggable notion of the architecture. The architecture takes care, that each of the entry points is triggered to serve the request accordingly. This so-called *inversion of control* limits the effort of the service provider implementing the specific geoprocess model and ensures that the added functionality (data handler or algorithm) is performed sufficiently.

An Algorithm Repository provides a list of available geoprocess models hosted by the WPS instance. It updates the GetCapabilities response document and DescribeProcess response documents. Additionally, it forwards any Execute request to the specific algorithm. The Algorithm Repository is managed by the Repository Manager, which creates different Algorithm Repositories for different legacy systems. To manage built-in algorithms, the Local Algorithm Repository is installed by default.

The Repository Manager is designed to incorporate different legacy systems such as GRASS or SEXTANTE (see Section 2.4). It initializes for each of the legacy systems an Algorithm Repository and forwards the calls accordingly. Thereby it is possible to incorporate the full set of functionality of a specific legacy system without incorporating the functionality on an individual basis (i.e. per single geoprocess model).

The Data Handler Repository lists all available data handlers and forwards any input and output parameter containing geodata to the applicable data handler. The data handler has to convert the geodata from the external format to an internal object representation. The object representations of geodata are based on the GeoTools library.

3.2 Example of a WPS Client Application

Client applications to configure and perform distributed geoprocessing are required to enable user-friendly interaction with such kind of services. Several client applications have been developed as extensions of GvSIG, OpenLayers, JUMP and uDig. For this chapter, we will illustrate the WPS client application realized as a plug-in for uDig. This plug-in is used throughout the chapter (e.g. Section 3.3 and Section 4.1), as a reference implementation. uDig has been chosen to be the suitable candidate for extending it with capabilities for communicating with WPS, as it already allows users to integrate distributed data available through Web Services.

The client is realized as a plug-in in uDig, which is organized as a set of plug-ins on top of the eclipse rich client platform (Clayberg & Rubel, 2008). The eclipse RCP implements the concept of inversion of control and allows thereby only specific parts of the workflow to be customized. The customization within the eclipse RCP is achieved via so called extension points, at which the intended functionality can be inserted into the RCP framework. The concept of extension points is inherited by uDig to insert intended functionality at designated points of the uDig workflow.

To the user, the WPS client plug-in appears to be a Wizard, in which the user can inspect, configure and perform the geoprocess model. During the configuration, all the data which is available to the client may serve as input to the specific geoprocess model. Like all the data available in uDig, the final result of the geoprocess model is available as a layer and can serve as input for further processing.

The user interaction with the wizard is depicted in Figure 8. In the given example, a road data set served by WFS needs to be simplified using Douglas-Peucker algorithm. First the designated type of service has to be chosen - in this case WPS (Step I.). The user is now able to register the entry point of the WPS instance (i.e. URL) (Step II). Based on this URL, the client is able to retrieve the GetCapabilities document of the specific WPS instance automatically and retrieves all the available process descriptions,

which the user can inspect in Step III. The user is now able to select the designated geoprocess model (i.e. Douglas-Peucker algorithm). Based on the selected geoprocess model and its process description, the wizard generates a form with the required parameters, which have to be supplied by the user (Step IV.). For every required literal input, the user is presented with a corresponding text box. Additionally, for every complex input, the user can choose from a drop down list containing currently available layers (i.e. registered geodata) inside of uDig. If the user chooses a WFS layer, the layer can be send by reference according to the general wizard configuration. This allows the WPS to fetch the data and saves up time, since not the whole data has to be prepared on the client-side and transferred to the service. In the given example, the user provides a tolerance value for the Douglas-Peucker algorithm and a road dataset, which is sent by reference to the WPS, as it is provided through a remote WFS instance. As a geoprocess model might provide multiple results, the user can select the appropriate ones, which need to be added as layers in uDig (Step V.).

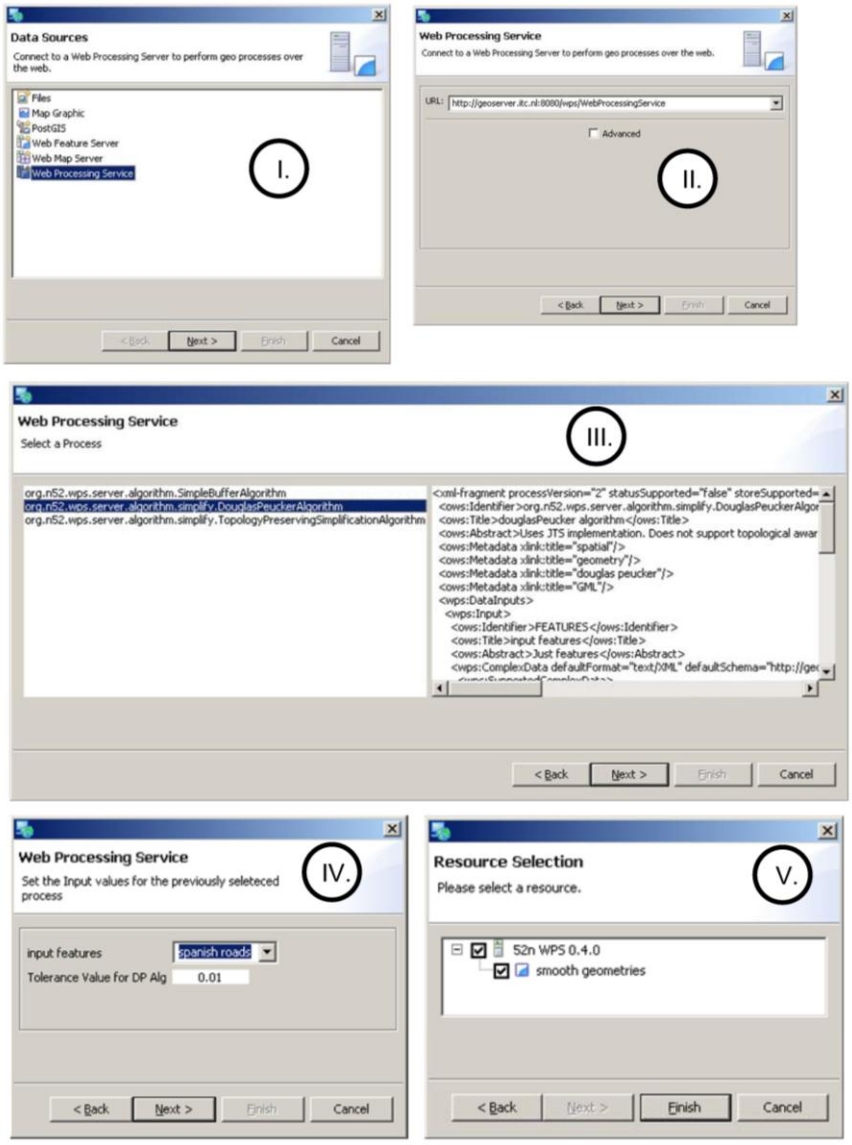


Figure 8: User interaction with wizard to perform Douglas-Peucker algorithm hosted by WPS (Foerster & Schaeffer, 2007).

The final result is presented as a separate layer in uDig as shown in Figure 11 (top).

3.3 Geoprocessing Services and Geospatial Mass-market Applications

Geospatial mass-market applications such as Google Earth allow ordinary user to access and share geospatial information over the web in an intuitive way. Mass-market applications provide full coverage of satellite base data and mechanisms to integrate geospatial information available on the web.

Geoprocessing Services provide real-time geoinformation and users demand real-time geoinformation especially in cases of emergency (e.g. flooding or forest fires). Thereby, the integration of such web-based information is promising for current geospatial mass-market applications to provide the users comprehensive access to up-to-date information.

The basic way, how geospatial information is described and exchanged in mass-market applications, is KML. KML is widely used in applications such as Google Maps and Google Earth and became an official OGC standard (OGC, 2008). KML is unique in the family of OGC data encodings, as it combines data encoding, styling and the special network facilities, which are called *NetworkLinks* and are also known as dynamic KML. These NetworkLinks are especially interesting for web-based geoinformation, as they allow the dynamic integration of remote resources (e.g. provided by WPS). Therefore, the content of a KML file might become dynamic and provide temporal data (e.g. from sensors). As NetworkLinks use URLs, KML is not only bound to file-based access, but can link any Web Service, as long as it operates via HTTP-GET and serves KML. NetworkLinks provide additional properties such as update, scheduling etc.

This section will describe the approach to integrate web-based geoinformation generated by WPS-based geoprocess models into such geospatial mass-market applications. Finally, the chapter will demonstrate the applicability of the proposed approach by a fire threat use case. This use case demonstrates how ordinary users can access the most current information through their geospatial mass-market application and take actions accordingly.

3.3.1 The Approach of Integrating WPS and Geospatial Mass-market Applications

To integrate WPS-based geoprocess models, several requirements of the geospatial mass-market applications have to be met. The major requirement is that the communication pattern (REST architecture & KML encoding) of the geospatial mass-market applications does not have to be changed. Thereby, the WPS-based geoprocess models become capable of being seamlessly integrated into such applications. This requirement is met by the WPS interface specification, as it supports the invocation of geoprocess models via HTTP-GET, which is a common approach of REST-architectures. Additionally, the WPS interface specification does not foresee any data encoding for its input and output parameters, thus KML is a valid format for the WPS. Finally, as the WPS is able to return process results as raw data without any WPS-specific message overhead it is highly applicable for an integration into geospatial mass-market applications.

However, as the configuration of such process models is highly complex and not supported by current geospatial mass-market applications, a twofold approach is proposed. At first, the expert user selects and configures the geoprocess model through an expert user interface, most-likely a GIS. At second, the user

integrates the pre-configured geoprocess model into his/her geospatial mass-market application of choice. This integration is possible, as the WPS interface meets the requirements of geospatial mass-market applications, as explained in the previous paragraph. The pre-configuration eases the integration of such geoprocess models and thereby the integration of web-based geoinformation in Google Earth for a non-expert user and is thereby not a drawback. Moreover, it prevents from false configuration of the geoprocess model through the non-expert user. This sequence of action is also depicted in Figure 9.

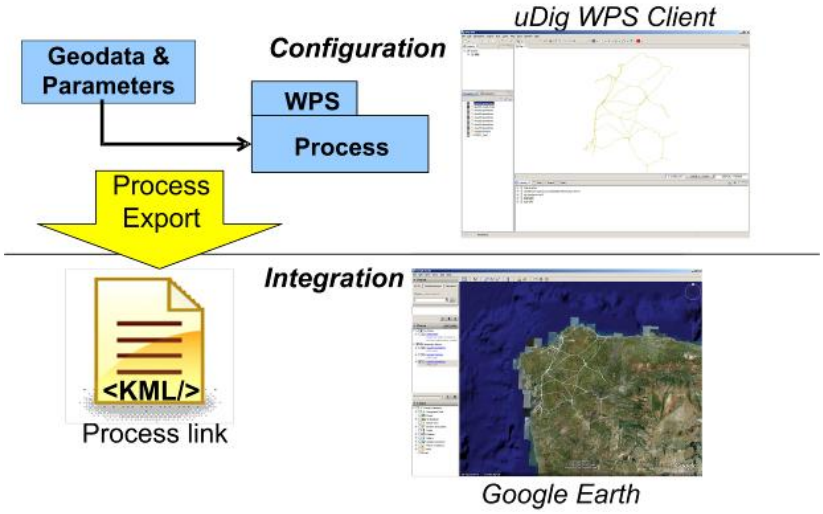


Figure 9: Approach to integrate WPS-based geoprocess models in mass-market applications such as Google Earth.

In particular, the WPS-based geoprocess models are configured through uDig (Section 3.2). For this approach, uDig has been enhanced to export these configured WPS-based geoprocess models as KML.

The export of the configured geoprocess model from uDig to KML can be configured in two ways:

1. Export the KML file as a link to a stored process result. This is the *static* option, in which no geoprocess model will be triggered when visualizing the KML file in Google Earth. This uses the store functionality of the WPS interface specification.
2. Export the KML file as a link, which triggers the WPS-based geoprocess model. This is the *dynamic* option and enables to trigger the geoprocess model in real-time, when incorporating the file in Google Earth. This allows one also to set a refresh rate to initiate performing the geoprocess model on the server again. It is important to note, that in this case, the WPS-based geoprocess model is triggered and if any WPS input data is defined as reference, the (updated) data is fetched and used as the basis for the calculation. This approach allows the processing of the latest available data and thus visualizing the latest results in mainstream applications.

In both cases the files incorporate the links using the *NetworkLink* functionality of KML. Figure 10 depicts the GUI dialog in uDig to configure the KML file referencing the configured geoprocess model (regarding the applied process strategy).

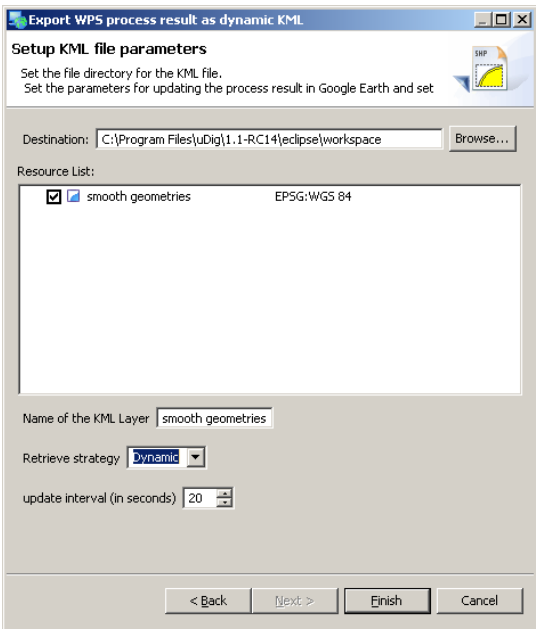


Figure 10 User interface to export the configured geoprocess model in uDig as KML.

Listing 9 shows the generated NetworkLink using the dynamic option in the KML export of uDig (option 2). The generated KML includes an Execute request via HTTP-GET to a Douglas-Peucker algorithm for simplification, which is also used in the scenario described in Section 3.3.2. The request references remote WFS data.

Listing 9: KML NetworkLink with a WPS-Execute request via HTTP-GET. The request references remote data hosted on WFS.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
  <Folder>
     <name>smooth geometries</name>
  <visibility>0</visibility>
  <open>0</open>
  <description>WPS Layer</description>
    <NetworkLink>
          <name>WPS Layer</name>
          …
          <description>WPS Layer</description>
          <refreshVisibility>0</refreshVisibility>
           <Link>
             <href>http://geoserver:8080/wps/WebProcessingService?request=execute&amp;service=WPS&amp;version=1.0.0&amp;Identifier=org.n52.wps.server.algorithm.simplify.DouglasPeuckerAlgorithm&amp;DataInputs=FEATURES=@mimeType=text/xml@href=http%3A%2F%2Fgeoserver%3A8080%2Fgeoserver%2Fwfs%3FSERVICE%3DWFS%26VERSION%3D1.0.0%26REQUEST%3DGetFeature%26typename%3Dtopp%3Aspanish_roads@Schema=http://schemas.opengis.net/gml/2.1.2/feature.xsd;TOLERANCE=1&amp;RawDataOutput=SIMPLIFIED_FEATURES@mimeType=application/vnd.google-earth.kml%2Bxml@schema=http://www.opengis.net/kml/2.2</href>
  <refreshMode>onInterval</refreshMode>
  <refreshInterval>20</refreshInterval>
   …
</kml>
```

By supporting these two options (dynamic vs. static), the integration is well-scalable and applicable to scenarios requiring dynamic or even static process results.

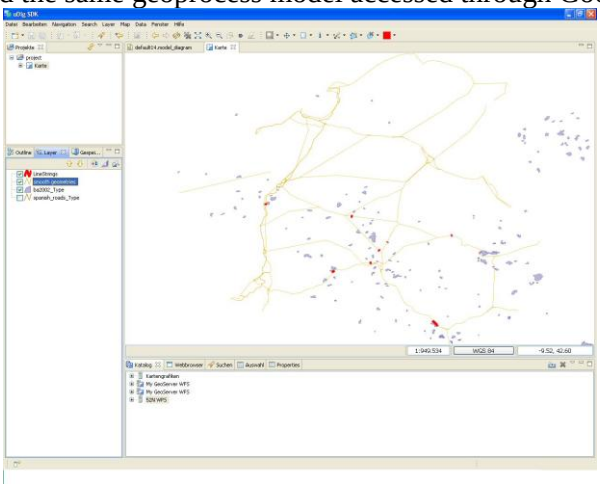
Finally the exported KML files are loaded into Google Earth. The result of the geoprocess model is then displayed directly with imagery and other content available in Google Earth.

It is important to note, that the client application (uDig) is able to perform and export single and chained geoprocess models and geoprocessing workflows. This task of chaining is performed locally at the client side and does not involve any central processing engine. A more sophisticated approach for geoprocessing workflows by the means of BPEL is described in Section 2.2.

3.3.2 Integrating Mass-market Applications for a Risk Management Scenario

The scenario is settled in the context of a risk management use case, in which in-situ-sensor data has to be analyzed for assessing a fictive fire threat in Northern Spain. The scenario and the involved services have been extensively presented in (Foerster & Bastian Schaeffer, 2007). In this example, the services and data are taken from the ORCHESTRA project, which addresses a similar fire risk management scenario (Friis-Christensen et al., 2006).

In the given scenario the expert configures the geoprocess model in the WPS client plug-in for uDig by buffering the fire threat areas and intersecting them with the road data. The buffering operation and the intersection operation are single WPS-based geoprocess models. The road data has been additionally simplified to improve the process performance and to improve portrayal at smaller scales. Overall, this allows the expert to assess which parts of the road infrastructure are at risk by a fire threat. The expert user exports the configured geoprocess model as a KML file and links it on the national portal site. The citizen (i.e. non-expert user) is now able to visualize the latest analysis results regarding the affected roads in his/her geospatial mass-market application by loading the KML file from the portal site. He/she can inspect the latest geoinformation (as calculated by a specific WPS-based geoprocess model) with underlying base information from areal imagery and/or topography and other content such as available in Google Earth. In the given scenario the specific citizen can inspect the current status of the road infrastructure and is able to perform appropriate actions. Figure 11 depicts the result of the configured geoprocess model in uDig and the same geoprocess model accessed through Google Earth.



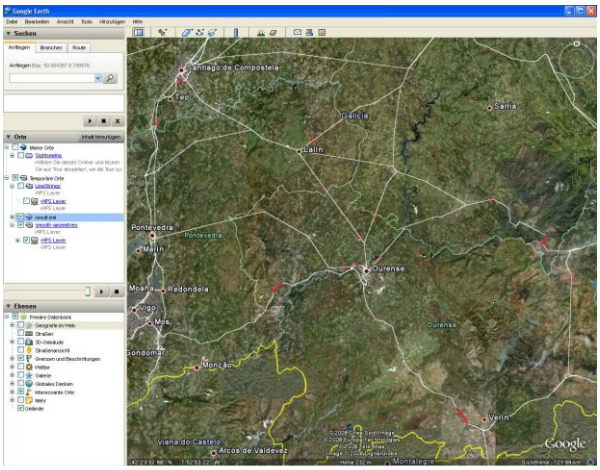


Figure 11: Screenshots of the configured geoprocess model in uDig (top) and exported to Google Earth (bottom) – simplified roads & affected road infrastructure (red).

4 Scenarios

This section examines two different scenarios to demonstrate the concepts of WPS, geoprocessing workflows, QoS and legacy systems as introduced in Section 2. One scenario focuses on an analysis of particulate matter distribution in Germany for air quality monitoring. This scenario involves different types of geoprocess models combined into a workflow. The other scenario presents the applicability of Grid and Cloud Computing as an application for increasing processing power for a risk management scenario in Taiwan.

4.1 Air Quality Monitoring in Germany

This section presents a real-world scenario, which illustrates the presented concepts (Section 2). Ambient air quality has been identified by the European Commission as one of the most critical environmental aspects. Therefore, it is one of the major objectives in the Sixth Environmental Action Programme[4]. Through legislation, the European Union has also been active in this field for several decades. With the Council Directives 1999/30/EC and 1996/62/EC strict air quality limits for several air quality parameters were specified, which have to be adapted by national law by all member states.

This scenario addresses Particulate Matter (PM10) as one of the most hazardous aerosols. Several studies have researched the correlation of PM10 and health effects. For instance, Pope et al. (2002) found that for each rise of 10 $\mu g/m^3$ the lung cancer rate increases by 4-8%. Hence, the real-time monitoring of PM10 is a crucial task to support decision makers in protecting public health and to comply with the EU regulations. To automate this business process, a model realized as a geoprocessing workflow will be developed to monitor real-time spatial distributions of PM10 by applying the concepts of distributed geoprocessing.

Figure 12 presents the deployed workflow. It was modeled with the 52°North Workflow Modeler, which automatically creates WSDL documents for the workflow partners and describes the workflow via BPEL (Bastian Schaeffer & Foerster, 2008). While for instance Weiser & Zipf (2007) also showed the orchestration of OGC Web Services with BPEL, they had to build the WSDL manually. Our approach goes beyond that and exposes the deployed workflow as a simple WPS-based geoprocess model using the

[4] The Sixth Environment Action Programme of the European Community: http://ec.europa.eu/environment/newprg/index.htm.

WPS-T approach (Section 2.2). The client on the left-hand side (Figure 12) is now capable to invoke the workflow like any other WPS-based geoprocess model (Step 1). Only the reference to the sensor data has to be included in the request. In our case, the 52°North WPS uDig client application (Section 3.2) can be used to invoke the service and visualize the results (Figure 13).

After invoking the workflow represented by a WPS-based geoprocess model, hosted on the WPS-T, the underlying workflow engine is triggered in the backend (Step 2). Upon then, the workflow engine orchestrates the workflow and invokes each of the workflow partners. At first, a WPS-based filter process, which takes a URL from an OGC Sensor Observation Service (SOS) as input (Step 3) is triggered. The SOS instance is located on top of a sensor network and allows the client to query the observed phenomena in a standardized format. In this scenario real-time air quality data from the state of North-Rhine Westphalia, located in the western part of Germany, is available through the SOS instance. The SOS instance provides up-to-date data of 56 monitoring stations distributed throughout this region. Internally, the WPS queries the SOS for its latest available sensor data regarding this specific region (Step 4). Next, the retrieved sensor data is converted into the common Geography Markup Language (GML) format (Step 5), which can be used for further processing. In particular, the geometry and observed values are extracted and new GML point features are created. This geoprocess model is performed by a Grid Computing enabled WPS, which delegates the conversion task to its underlying grid middleware due to high computational complexity. The following step (Step 6) takes the resulting point data from the second geoprocess model as input and interpolates them with the GRASS Inverse Distance Weighting (IDW) method to a resulting GeoTIFF raster. In the last step (Step 7) the GeoTIFF raster becomes visualized according to the GRASS color scheme for air pollution is applied, in which the color red represents values above a specific threshold (40 µg PM10 /m³ air as the current EU legislation limit). The output is visualized as a new layer in uDig (Figure 13).

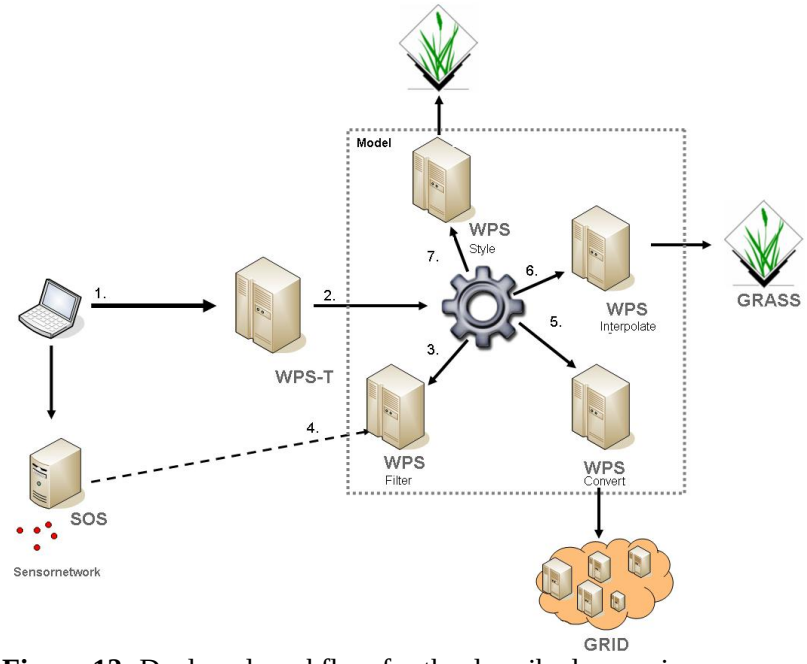


Figure 12: Deployed workflow for the described scenario.

As the workflow analyses latest data in (almost) real-time and can be triggered on-demand, it allows the decision makers to monitor the current air quality and thereby to take actions accordingly based on the latest information. Additionally, it is possible to compare the phenomena over time and thereby get analyze its behavior more thoroughly.

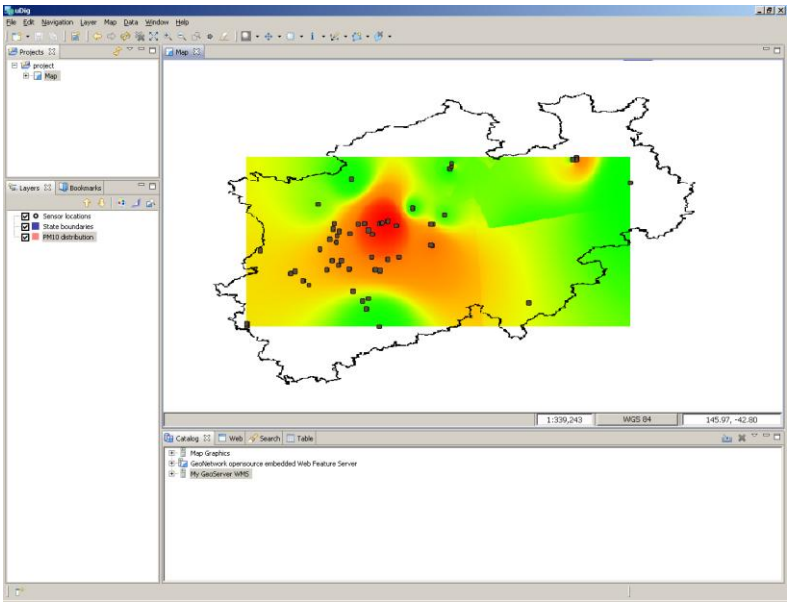


Figure 13: Workflow results visualized with states boundary of North-Rhine Westphalia and the sensor location. The red colored areas imply a high amount of PM10 (40 µg PM10 /m³ air), which violates current EU regulations.

4.2 Grid Computing for Risk Management in Taiwan

Taiwan is located at the collision boundary of the Philippine Sea plate and the Eurasian plate. The mountain terrain is precipitous and the region, on the whole, is characterized by fragile rocks and frequent seismic activity. In addition, the concentrated torrential rainfall brought by typhoons causes extensive disasters. A debris flow, the most serious disaster caused by torrential rainfall, leads to very heavy casualties in recent years. Since 2002, the Soil and Water Conservation Bureau, which is responsible for the conservation and administrative management of hillside in Taiwan, has been cooperated with the Feng Chia University to build up a national debris flow monitoring system. Together they have successively carried out the establishment and maintenance of 13 fixed debris flow monitoring stations over the island. The main concept of the debris flow monitoring system is to connect various sensors (e.g. satellites, fixed and mobile monitoring stations, remotely operated aircrafts, etc.) to establish a network of ubiquitous monitoring to command, control, communicate, survey and reconnoiter to give intelligence to decision makers.

However, the whole architecture was designed in late 2000 and implemented by traditional monolithic and proprietary methodologies. Since then, several interoperability issues have been unveiled in the recent years. An open standards-based architecture for debris flow monitoring following the SDI paradigm was presented in Chung, Fang, Chou, B. Lee, & Baranski (2009). The architecture applies OGC standards such as the specification of the Sensor Web Enablement (SWE) to connect to the deployed sensors and the WPS interface specification for processing the delivered sensor information.

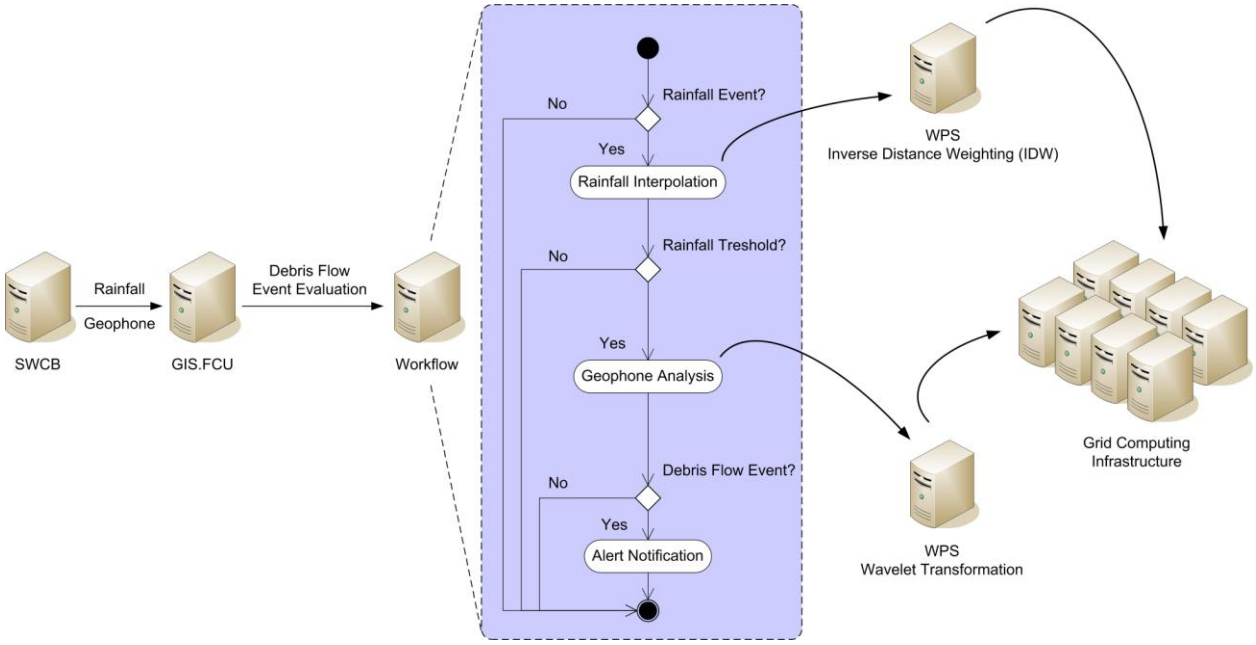


Figure 14: An overview about the OWS-6 GPW Debris Flow demonstration scenario workflow.

During typhoon season, the debris flow monitoring system operates under full load and a lot of data from many sensors is produced and must be analyzed. In the OWS-6 testbed a proof of concept was developed, that utilizes a Grid Computing infrastructure for increasing the performance of processing the large amounts of live sensor data streams. An overview about the OWS-6 debris flow demonstration scenario workflow is given in Figure 14. The developed *Rainfall Interpolation* process implements an Inverse Distance Weighting (IDW) algorithm and calculates the rainfall distribution in a specific region depending on several rain gauges deployed at a fixed debris flow monitoring station. The developed *Geophone Analysis* process implements a wavelet transformation algorithm and calculates if the measured ground vibration reaches a specific threshold. Both processes use the Grid Computing enabled 52°North WPS framework (Figure 15) and split the processes task into smaller sub-tasks that will be performed distributed in a Grid Computing infrastructure based on the grid middleware UNICORE (Huber, 2001) (http://www.unicore.eu/).

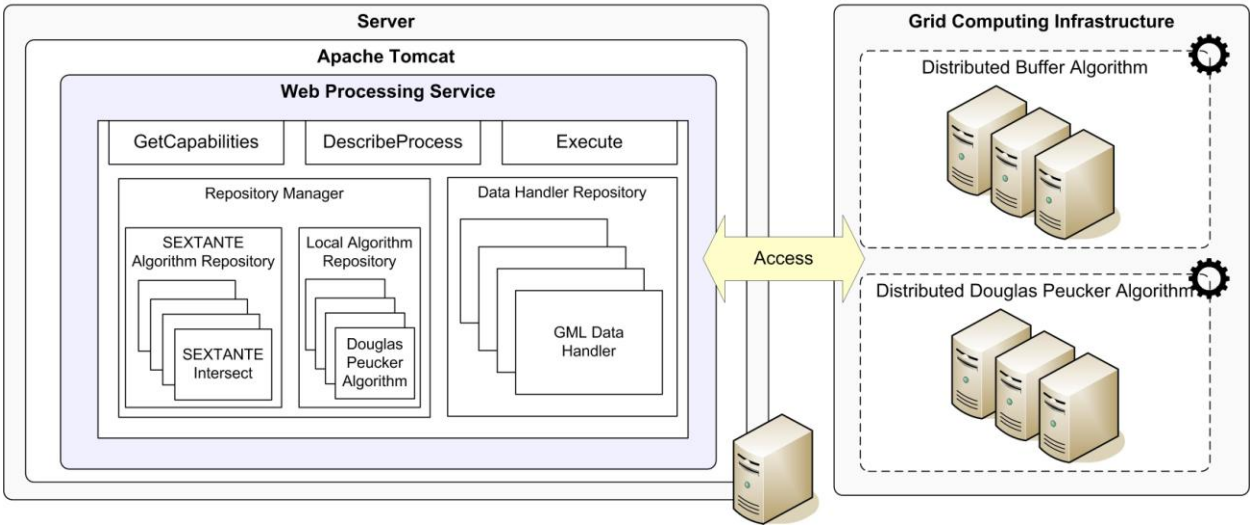


Figure 15: Architecture of the Grid Computing enabled 52°North WPS framework.

If it begins to rain, a decision maker has to analyze if a debris flow event will be happen or not. The decision maker sends a request to a pre-defined workflow engine (realized through BPEL). The workflow engine requests the database for the measured rain gauge and earth vibration data. Afterwards, the workflow invokes the Rainfall Interpolation process, which runs in parallel on a Grid Computing enabled WPS to increase the computational performance. If the rainfall reaches a specific threshold, the workflow invokes the (also distributed and parallel) Geophone Analysis process. If the computation indicates a debris flow event, the workflow will send an alert notification via mail to a decision maker.

5 Conclusion

This chapter presents current approaches for establishing distributed geoprocessing on the Web. It thereby shows how SDIs can be technically enhanced to provide geoinformation using the existing web-based data sources (e.g. provided by WFS or WCS). SOA and Geospatial Web Services are essential for the interoperability of web-based geoprocess models in SDIs. The WPS interface specification is presented as the OGC-compliant approach enabling single geoprocess models in SOAs. Single geoprocess models can be chained to complex ones using geoprocessing workflows. Further, these geoprocessing workflows can be deployed as single geoprocess models using WPS-T (Section 2.2.2). This approach in particular combines the expressiveness of existing workflow technologies of mainstream IT (e.g. BPEL) and existing OGC interface specifications (i.e. WPS). The presented approaches to ensure QoS within SOAs support distributed geoprocessing on an operational level. The integration of legacy systems contributes to its functional level.

Many projects are dedicated to distributed geoprocessing for geo-related decision support. Based on distributed geoprocessing, decisions can benefit from the most current data retrieved from different locations and transformed to suitable geoinformation using various complex geoprocess models. Specific applications are necessary to publish and access the geoprocess models. For a framework implementing the WPS interface specification, it is important to be extensible to limit the effort of integrating and adopting it into existing SOAs. This is demonstrated by the example of 52°North WPS framework. The client applications have to meet specific user requirements, as examined by the WPS plug-in for uDig and the mass-market application. In the latter case, the architecture had to meet the requirements of the mass-market application (i.e. KML and NetworkLinks) for integrating geoprocess models (i.e. the generated geoinformation).

To demonstrate the introduced concepts and applications, Section 4 illustrates how real-world scenarios can be addressed. As already mentioned throughout the chapter, some research challenges still remain.

The automation of distributed geoprocessing and its performance are still unsolved. The automated creation of geoprocessing workflows is still lacking due to missing semantics. These semantics need to be included in the description of the geoprocess models and the geoprocessing workflows itself. Therefore thorough classifications are required, which might be modeled as ontologies. Based on these semantics it will be possible to enable reasoning for automatically selecting single geoprocess models and chaining them to geoprocessing workflows.

Reliability and performance of distributed geoprocessing also require further concepts. For example, the current WPS specification does not offer any possibility to list the running geoprocess models at a WPS instance, to prioritize geoprocess models, to interactively pause and resume, nor to stop and restart geoprocess models. Furthermore, the current WPS specification does not give the users of a WPS any enhanced control over the computational resources in the backend (e.g. for advanced resource reservation and corresponding guaranteed provision of requested resources). However, not only standardization is important, but also research on the specific geoprocess model is required by optimizing the implementation or finding new algorithms.

Finally, solving the problems regarding the automation, reliability and performance of distributed geoprocessing will increase its applicability and will show the benefit of integrating functionality and data from distributed resources to efficiently support decision making and thereby support the future vision of SDI as stated for instance in Williamson, Rajabifard, & Feeney (2003).

References

van der Aalst, W., Dumas, M., Hofstede, A. T., Russell, N., Verbeek, H., & Wohed, P. (2005). Life After BPEL? In *Formal Techniques for Computer Systems and Business Processes*, Lecture Notes in Computer Science (Vol. 3670, pp. 35-50). Berlin, Germany: Springer.

Ackoff, R. (1989). From data to wisdom. *Journal of Applied System Analysis*, *16*, 3-9.

Alameh, N. (2003). Chaining Geographic Information Web Services. *IEEE Internet Computing*, *07*(5), 22-29. doi:10.1109/MIC.2003.1232514

Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2004). *Web Services* (1. ed.). Springer Verlag.

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., et al. (2003). *Business process execution language for Web Services* (No. version 1.1) (p. 136). OASIS. Retrieved from

http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf

Anjomashoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., & Pulsipher, D. (2005). *Job Submission Description Language (JSDL)* (Specification No. GFD-R.056). Global Grid Forum.

Baranski, B. (2008). Grid Computing Enabled Web Processing Service. In E. Pebesma, M. Bishr, & T. Bartoschek (Eds.), *Proceedings of the 6th Geographic Information Days*, IfGI prints (Vol. 32, pp. 243-256). Presented at the GI-days 2008, Muenster, Germany: Institute for Geoinformatics. Retrieved from http://www.gi-tage.de/archive/2008/downloads/acceptedPapers/Papers/Baranski.pdf

Baranski, B. (2009). *OWS-6 WPS Grid Processing Profile Engineering Report* (OGC Public Engineering Report No. 09-041r3) (p. 101). OGC.

Baranski, B., Schaeffer, B., & Redweik, R. (2009). Geoprocessing in the Clouds. In *Proceedings of Free and Open Source Software for Geospatial Conference*. Presented at the Foss4g 2009, Sydney, Australia: OSGeo.

Bernard, L., Craglia, M., Gould, M., & Kuhn, W. (2005). Towards an SDI Research Agenda. In *Proceedings of the 11th EC_GIS Conference* (pp. 147-151). Presented at the 11th EC GIS & GIS Workshop - ESDI: Setting the Framework, Sardinia.

Brauner, J. (2008). Providing GRASS with a Web Processing Service Interface. In E. Pebesma, M. Bishr, & T. Bartoschek (Eds.), *Proceedings of the 6th Geographic Information Days*, IfGI prints (Vol. 32, pp. 91-116). Presented at the GI-days 2008, Muenster, Germany: Institute for Geoinformatics.

Brauner, J., Foerster, T., Schaeffer, B., & Baranski, B. (2009). Towards a Research Agenda for Geoprocessing Services. In J. Haunert, B. Kieler, & J. Milde (Eds.), *12th AGILE*

International Conference on Geographic Information Science. Presented at the AGILE 2009, Hanover, Germany: IKG, Leibniz University of Hanover. Retrieved from http://www.ikg.uni-hannover.de/agile/fileadmin/agile/paper/124.pdf

Brauner, J., & Schaeffer, B. (2008). Integration of GRASS functionality in web based SDI service chains (pp. 420-429). Presented at the FOSS4G 2008, Cape Town, South Africa. Retrieved from http://www.osgeo.org/ocs/index.php/foss4g/2008/paper/view/133

Bucher, B., & Jolivet, L. (2008). Acquiring service oriented descriptions of GI processing software from experts. In L. Bernard, A. Friis-Christensen, H. Pundt, & I. Compte (Eds.), *AGILE 2008*.

Cardoso, J., & Sheth, A. (2003). Semantic E-Workflow Composition. *Journal of Intelligent Information Systems*, *21*(3), 191-225. doi:10.1023/A:1025542915514

Cepicky, J., & Becchi, L. (2007). Geospatial processing via Internet on remote servers - PyWPS. *OSGeo Journal*, *1*, 5p.

Chen, M., Ebert, D., Hagen, H., Laramee, R., van Liere, R., Ma, K., Ribarsky, W., et al. (2009). Data, Information, and Knowledge in Visualization. *Computer Graphics and Applications, IEEE*, *29*(1), 12-19. doi:10.1109/MCG.2009.6

Chung, L., Fang, Y., Chou, T., Lee, B., & Baranski, B. (2009). A SOA based debris flow monitoring system. Architecture and proof-of-concept implementation. In *17th International Conference on Geoinformatics* (pp. 1-6). Presented at the Geoinformatics 2009, Washington, USA: IEEE. doi:10.1109/GEOINFORMATICS.2009.5293549

Clayberg, E., & Rubel, D. (2008). *Eclipse Plug-ins* (3. ed.). Addison-Wesley Professional.

Di, L., Chen, A., Yang, W., & Zhao, P. (2003). The Integration of Grid Technology with OGC Web Services (OWS) in NWGISS for NASA EOS Data (pp. 24-27). Presented at the

GGF8 & HPDC12 2003, Seattle, WA, USA: Science Press.

Diaz, L., Costa, S., Granell, C., & Gould, M. (2007). Migrating geoprocessing routines to web services for water resource management applications. In M. Wachowicz & L. Bodum (Eds.), *10th AGILE International Conference on Geographic Information Science*. Aalborg University, Denmark.

Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, *10*(2), 112-122.

Erl, T. (2005). *Service-Oriented Architecture : Concepts, Technology, and Design*. Prentice Hall PTR.

Foerster, T., Burghardt, D., Neun, M., Regnauld, N., Swan, J., & Weibel, R. (2008). Towards an Interoperable Web Generalisation Services Framework - Current work in progress. In *Proceedings of the 11th ICA workshop on generalization and Multiple representation*. Montpellier, France.

Foerster, T., & Schaeffer, B. (2007). A client for distributed geo-processing on the web. In G. Tayler & M. Ware (Eds.), *W2GIS*, Lecture Notes in Computer Science (Vol. 4857, pp. 252-263). Springer. doi:10.1007/978-3-540-76925-5

Foerster, T., & Stoter, J. E. (2006). Establishing an OGC Web Processing Service for generalization processes. In *ICA workshop on Generalization and Multiple Representation*. Portland, Oregon, USA. Retrieved from http://aci.ign.fr/Portland/paper/ICA2006-foerster_stoter.pdf

Friis-Christensen, A., Bernard, L., Kanellopoulos, I., Nogueras-Iso, J., Peedell, S., Schade, S., & Thorne, C. (2006). Building Service Oriented Application on top of a Spatial Data

Infrastructure - A Forest Fire Assessment Example. In *Proceedings of the 9th Agile Conference on Geographic Information Science* (pp. 119-127). Visegrad, Hungary.

Friis-Christensen, A., Lucchi, R., Lutz, M., & Ostlaender, N. (2009). Service chaining architectures for applications implementing distributed geographic information processing. *International Journal Of Geographical Information Science*, *23*(5), 561-580. doi:10.1080/13658810802665570

Friis-Christensen, A., Ostlander, N., Lutz, M., & Bernard, L. (2007). Designing Service Architectures for Distributed Geoprocessing: Challenges and Future Directions. *Transactions in GIS*, *11*(6), 799-818. doi:10.1111/j.1467-9671.2007.01075.x

Geraci, A. (1991). *IEEE Standard Computer Dictionary: Compilation of IEEE Standard Computer Glossaries*. (F. Katki, L. McMonegal, B. Meyer, J. Lane, P. Wilson, J. Radatz, M. Yee, et al., Eds.). Institute of Electrical and Electronics Engineers Inc., The.

Granell, C., Diaz, L., & Gould, M. (2007). Managing Earth observation data with distributed geoprocessing services. In *Geoscience and Remote Sensing Symposium* (pp. 4777 - 4780). Presented at the IGARSS 2007, IEEE. doi:10.1109/IGARSS.2007.4423928

Granell, C., Gould, M., & Ramos, F. (2005). Service composition for SDIs: integrated components creation. In *Proceedings of Sixteenth International Workshop on Database and Expert Systems Applications* (pp. 475-479). Presented at the Workshop on Geographic Information Management (DEXXA 05), Copenhagen: IEEE. doi:10.1109/DEXA.2005.179

Hartig, K. (2009). What is Cloud Computing? *Cloud Computing Journal, SYS-CON Media Inc.*

Henneboehl, K., & Pebesma, E. (2008). Providing R functionality through the OGC Web Processing Service. Presented at the The R user conference, Technische Universitaet

Dortmund.

Huber, V. (2001). UNICORE: A Grid Computing Environment for Distributed and Parallel Computing. In V. Malyshkin (Ed.), *Proceedings of the 6th International Conference on Parallel Computing Technologies* (Vol. 2127, pp. 258-265). Presented at the PaCT 01, Novosibirsk, Russia: Springer.

INSPIRE. (2007a). Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community. *Official Journal of the European Union*, 18.

INSPIRE. (2007b). *INSPIRE Network Services Performance Guidelines* (p. 22). INSPIRE Consolidation Team.

INSPIRE. (2008). *INSPIRE Network Services Architecture* (p. 30). European Commission.

ISO/TC 211. (2005). *Geographic information - Services* (ISO Standard 19119 No. ISO 19119) (p. 67). International Organization for Standardization.

ITU-T. (1994). *Telephone Network and ISDN - Quality of Service, Network Management and Traffic Engineering* (Chapter Terms and Definitions related to Quality of Service and Network No. Recommendation E.800). International Telecommunication Union (ITU).

ITU-T. (2001). *Transmission Systems and Media, Digital Systems and Networks* (No. Recommendation G.1000). International Telecommunication Union (ITU).

de Jesus, J., Hiemstra, P., & Dubois, G. (2008). Web-based geostatistics using WPS. In *Proceedings of GI-days 2008,* Ifgi-prints. Institute for Geoinformatics.

Keens, S. (2007). *OWS-4 Workflow IPR* (OGC IPR No. 06-187). OGC.

Kiehle, C., Greve, K., & Heier, C. (2006). Standardized Geoprocessing - taking spatial data infrastructures one step further. In *9th AGILE International Conference on Geographic*

Information Science (pp. 273-282). Visegrad, Hungary.

Kiehle, C., Heier, C., & Greve, K. (2007). Requirements for Next Generation Spatial Data Infrastructures-Standardized Web Based Geoporcessing and Web Service Orchestration. *Transactions in GIS, 11*(6), 819-834. doi:10.1111/j.1467-9671.2007.01075.x

Lake, R., & Farley, J. (2007). Infrastructure for the Geospatial Web. In A. Scharl & K. Tochtermann (Eds.), *The Geospatial Web,* Advanced Information and Knowledge Processing Series (pp. 15-26). London, UK: Springer.

Lanig, S., Schilling, A., Stollberg, B., & Zipf, A. (2008). Towards Standards-based Processing of Digital Elevation Models for Grid Computing through Web Processing Service (WPS). In *ICCSA*, Lecture Notes in Computer Science (Vol. 5073, pp. 191-203). Presented at the Computational Science and Its Applications - ICCSA 2008, Perugia, Italy: Springer Verlag. doi:http://dx.doi.org/10.1007/978-3-540-69848-7_17

Lee, J., Jeon, W., Lee, W., Jeong, S., & Park, S. (2003). *QoS for web services: Requirements and possible approaches*. W3C. Retrieved from http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/

Lehto, L. (2007). Schema Translations in a Web Service Based SDI. In M. Wachowicz & L. Bodum (Eds.), *10th Agile International Conference on Geographic Information Science 2007*. Aalborg University, Denmark.

Lemmens, R. (2006). *Semantic interoperability of distributed geo-services* (PhD thesis). Delft University of Technology. Retrieved from http://www.ncg.knaw.nl/Publicaties/Geodesy/pdf/63Lemmens.pdf

Lutz, M. (2007). Ontology-based Descriptions for Semantic discovery and composition of Geoprocessing Services. *GeoInformatica, 11*(1), 1-36. doi:10.1007/s10707-006-7635-9

Müller, M., Vogel, R., & Bernard, L. (2009). Multi-Criteria Evaluation for Emergency Management in a Web Service Environment. In M. Konecny, S. Zlatanova, T. Bandrova, & L. Friedmannova (Eds.), *Cartography and Geoinformatics for Early Warning and Emergency Management: Towards Better Solutions* (pp. 439-446). Presented at the Joint Symposium of ICA Working Group on CEWaCM and JBGIS Gi4DM, Prague, Czech Republic: Masaryk University Brno.

Myerson, J. (2008). *Cloud computing versus grid computing* (p. 9). IBM Corporation. Retrieved from http://www.ibm.com/developerworks/web/library/wa-cloudgrid/

Nash, E. (2008). WPS Application Profiles for Generic and specialised Processes. In E. Pebesma, M. Bishr, & T. Bartoschek (Eds.), *Proceedings of the 6th Geographic Information Days*, IfGI prints (Vol. 32, pp. 69-79). Presented at the GI-days 2008, Muenster, Germany: Institute for Geoinformatics.

Neteler, M., & Mitasova, H. (2008). *Open Source GIS: A GRASS GIS Approach*. The International Series in Engineering and Computer Science (3. ed., Vol. 773). New Y: Springer.

Neun, M., Burghardt, D., & Weibel, R. (2008). Automated processing for map generalization using web services. *GeoInformatica*. doi:10.1007/s10707-008-0054-3

OGC. (2007). *OpenGIS Web Processing Service* (OGC implementation specification No. OGC 05-007r7). Open Geospatial Consortium. Retrieved from http://www.opengeospatial.org/standards/wps

OGC. (2008). *OGC KML* (specification No. OGC 07-147r2). OGC Standard (p. 251). Open Geospatial Consortium. Retrieved from https://portal.opengeospatial.org/files/?artifact_id=27810

Ostlaender, N. (2009). *Creating Specific Spatial Decision Support Systems in Spatial Data Infrastructures* (Phd thesis). University of Muenster.

Papazoglou, M. P., & Heuvel, W. (2007). Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal — The International Journal on Very Large Data Bases*, *16*(3), 389 - 415. doi:10.1007/s00778-007-0044-3

Peltz, C. (2003). Web services orchestration and choreography. *Computer*, *36*(10), 46-52. doi:10.1109/MC.2003.1236471

Ran, S. (2003). A model for web services discovery with QoS. *SIGecom Exchanges*, *4*(1), 1-10.

Roman, D., & Klien, E. (2007). SWING - A Semantic Framework for Geospatial Services. In A. Scharl & K. Tochtermann (Eds.), *The Geospatial Web*, Advanced Information and Knowledge Processing Series (pp. 229-234). London, UK: Springer.

Schaeffer, B. (2009). *OGC® OWS-6 Geoprocessing Workflow Architecture Engineering Report* (OGC Public Engineering Report No. 09-053r5) (p. 78). OGC.

Schaeffer, B., & Foerster, T. (2008). A Client for Distributed Geo-Processing and Workflow Design. *Journal for Location Based Services*, *2*(3), 194-210. doi:10.1080/17489720802558491

Scholten, M., Klamma, R., & Kiehle, C. (2006). Evaluating performance in spatial data infrastructures for geoprocessing. *IEEE Internet Computing*, *10*(5), 34-41.

Staab, S., van der Aalst, W., Benjamins, R., Sheth, A., Miller, J., Bussler, C., Maedche, A., et al. (2003). Web Services: Been There, Done That? *IEEE Intelligent Systems*, *18*(1), 72-85. doi:10.1109/MIS.2003.1179197

Stollberg, B., & Zipf, A. (2007). OGC Web Processing Service Interface for Web Service Orchestration - Aggregating Geo-processing Services in a Bomb Threat Scenario. In

Proceedings of Web and Wireless Geographical Information Systems, LNCS (pp. 239-251). Heidelberg: Springer-Verlag.

Stollberg, B., & Zipf, A. (2008). Geoprocessing Services for Spatial Decision Support in the Domain of Housing Market Analyses - Experiences from Applying the OGC Web Processing Service Interface in Practice. In L. Bernard, A. Friis-Christensen, H. Pundt, & I. Compte (Eds.), . Presented at the AGILE 2008, Girona, Spain.

Tu, S., Flanagin, M., Wu, Y., Abdelguerfi, A., Normand, E., Mahadevan, V., Ratcliff, J., et al. (2004). Design Strategies to Improve Performance of GIS Web Services. *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 04)*, *2*. doi:10.1109/ITCC.2004.1286692

Turton, I. (2008). GeoTools. In G. B. Hall & M. G. Leahy (Eds.), *Open source approaches in spatial data handlng*, Advances in geographic information (pp. 153-167). Berlin, Germany: Springer Verlag.

Vaughan-Nichols, S. (2002). Web Services: Beyond the hype. *IEEE Computer*, *35*(2), 18-21. doi:10.1109/2.982908

Weerawarana, S., Curbera, F., Leymann, F., Storey, T., & Ferguson, D. (2005). *Web Services Platform*

Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More. Prentice Hall PTR.

Weiser, A., & Zipf, A. (2007). Web Service Orchestration of OGC Web Services for Disaster Management. In J. LI, S. Zlatanova, & A. Fabbri (Eds.), *Geomatics Solutions for Disaster Management*, Lecture Notes in Geoinformation and Cartography (pp. 239-254). Berlin: Springer Verlag.

Werling, M. (2008). *OGC® OWS-5 GeoProcessing Workflow Architecture Engineering Report* (Discussion paper No. 07-138r1) (p. 34). OGC.

Williamson, I., Rajabifard, A., & Feeney, M. F. (2003). Future Directions in SDI Development. In *Developing Spatial Data Infrastructures: From concept to reality* (pp. 301-312). Taylor & Francis.