Providing GRASS with a Web Processing Service Interface

Johannes Brauner

Institute for Geoinformatics, University of Münster
joejoe@uni-muenster.de

Abstract. The process of bringing GIS functionality into a SOA based environment is at present undergoing its completion. Data maintenance and visualisation functionalities are readily available and commonly used in SDI. Nevertheless, one problem remains: SDIs do not offer real geoprocessing capabilities (as offered in classic desktop GIS) such as generalisation or complex raster based operations. Therefore, the OGC released the Web Processing Service Specification, which describes standardised abstract interfaces to bring complex processing power to SOA-based GIS. Despite this existing specification, hardly any concrete implementations of these abstract concepts can be found. Thus, it stands to reason to enhance a classic (desktop) GIS by a Web Processing Service interface to enable a full set of existing GIS functionalities. The paper describes two major aspects of how to achieve this by offering GRASS GIS functionalities through 52°North's WPS implementation. It is shown, that a semi-automatic approach to generate appropriate WPS interface descriptions for each GIS functionality is possible. Besides, it is described how to match the common statelessness of SOA services with the stateful charakter of a stand-alone GIS. This work is part of my M.Sc. thesis.

1 Introduction

A current trend in mainstream IT is to bring applications into a service oriented architecture (SOA) based environment (Blake et al. 2006). For details about the SOA concept see Erl (2005). Every functionality has to be offered as a web service instead of a desktop application. The geoinformation science community is influenced by this trend as well. To meet the community's more or less complex requirements, a “complete integration of geospatial data and geo-processing resources into mainstream computing” (Reed 2004 and various other OGC sources) as part of the Open Geospatial Consortium's (OGC) vision is aspired. To achieve this goal, OGC was founded in 1994 by players of the community and has released several abstract implementation specifications and standards for geodata representation and maintenance since.

After Burrough and McDonnell (1998, p. 11 ff.), a classic geographical information system (GIS) fulfils three main purposes:

- Data acquisition and maintenance,

- (Spatial) Analysis and modelling of geoobjects,
- Visualisation of geoinformation.

To achieve a full transition from GIS as a desktop application towards GIS accessible through a distributed system (in the GIS community such a system is called a spatial data infrastructure – SDI), each of the three GIS purposes has to be covered.

Due to the loose coupling of components or services inside a distributed system, only interfaces need to be standardised. Neither implementation details nor language nor execution platform is relevant for consumers (Chen et al. 2006). Therefore, for OGC's mission accomplishment (as described in its vision above) OGC has to standardise abstract interfaces and geodata formats.

According to OGC specifications and standards, data acquisition and maintenance is already supported by standards such as Simple Features for SQL or Geographic Markup Language (GML). For active data acquisition and generation, sensor data can be collected via standards from OGC's Sensor Web Enablement (SWE) initiative. (See OGC website for more information: http://opengeospatial.org)

Visualisation is as well and well covered by OGC efforts. There are numerous interface specifications for mapping services such as the Web Mapping Service (WMS) specification.

Even the combination and orchestration of different web services, the so-called service chaining, is possible. A mapping service can request vector data encoded in GML from a feature service, request another mapping service for an aerial view and combine this data into a map. The map can then be viewed online or printed by the user.

Although, one key feature is missing: The possibility to analyse geographic data, model geo-objects and enhance them to real geoinformation. To fill this gap, the OGC has just published the Web Processing Service (WPS) implementation specification (Open Geospatial Consortium 2007b). The specification provides interfaces to access any processing functionality inside a SDI. Eventually, this provides real geo-processing capabilities from a GIS point of view. Algorithms for generalisation, complex image processing (map algebra) and implementations of Egenhofer's operations (Egenhofer & Franzosa 1991) can now be enabled for on-line usage.

Even though WPS functionalities can be chained to complex work-flows (Schäffer 2007), no real implementations of geo-processing capabilities do exist. So, one problem still remains: How to make SDI capable of complex geo-processing? Papazoglou and van den Heuvel (2007) are suggesting to adapt and wrap old applications in SOAs. Is this applicable to SDIs and the WPS concept?

Simple WPS enabled solutions do exist (implementation of a single algorithm, a few libraries). But how can we bring a full set of functionalities e.g. the full functional range of a stand-alone desktop GIS (such as GRASS) to a SDI? This would bring geo-processing power to SDIs without reimplementing every single GIS functionality.

On-line usage could be accepted by users working with mobile devices in the field (for example emergency management) or occasional GIS users who could avoid buying and installing a complete GIS package on their own. Larger (or smaller) companies could prevent installing a GIS suite on every desktop PC. Further use cases and research are necessary to identify potential scopes of usage.

My paper is focussing on the automated generation of concrete GRASS process descriptions and how the stateless nature of OGC web services (OpenGIS Consortium 2002, p. 22) is overcome in regards to the stateful one of a monolithic desktop GIS. A prototypical implementation and much more details are part of my M.Sc. thesis.

2 Automated Generation of *DescribeProcess* Documents

WPS specification only provides abstract interfaces dealing with processes in general and not describing each individual GIS command or operation. Each process has to be described by a *DescribeProcess* XML document which has all information regarding to e. g. input- and output parameters, data formats, spatial reference system, and so on. Creating such a document is a rather complex process and can be an annoying task. Consider the amount of functionalities offered by a conventional GIS package (GRASS GIS has about 300 different operations). Hence, it would be convenient to accomplish this automatically.

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

Figure 1: Definition of an input layer for a buffer operation offered by GRASS interface description. (Abbreviated)

Fortunately, GRASS offers an XML based interface description for each command (command line parameter „--interface description“, see Figure 1), which is defined by an XML Document Type Definition (DTD). Following OGC OWS Common specification, all contents of OGC defined XML documents have to be specified by an XML Schema (Open Geospatial Consorti-

um 2007a, p. 82). Therefore, a structured XML content definition exists on both sides. This provides the basis for automated XML content transformation by an XSLT (Extensible Stylesheet Language Transformation) filter, which is capable of transforming structured XML contents from the GRASS interface description to a valid WPS DecribeProcess document (see Figure 2 for the transformed input layer from Figure 1). Implementing this filter was a rather complicated and time-consuming procedure, but excels writing hundreds of DescribeProcess documents by hand.

```
<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>
```

Figure 2: Definition of an input layer for a buffer operation inside a DescribeProcess document (abbreviated).

As a result, creating DescribeProcess documents automatically is possible. Unfortunately, two problems remain: semantics of input parameters and unnecessary processes.

First, a desktop GIS offers functionalities to visualise (e. g. opening a window and showing a 3D scene on the screen) and manage data, like copying layer XY to another workspace. These operations are not required for a web service based environment. Visualising data is platform dependent and part of the client side. Managing data is needless because WPS is a stateless service and therefore does not keep track of workspaces or (in GRASS terms) map locations. Every request has its own temporary workspace, which is deleted after creating a response and sending it back to the client. Thus, we need the know-how of a GIS expert to discard unimportant operations when creating WPS processes. In GRASS, it is relatively simple to disregard all visualisation commands, as they all start with a 'd.', like *d.monitor*. Handling of data management operations (usually starting with 'g.') is more difficult. Some of them are useless (e. g. location duplication), but some are not (changing projections with *g.proj* for example). For more information

about GRASS, consult Neteler & Mitasova (2007). Additional problems concerning stateless WPS nature are described in the following section.

Secondly, an XSLT filter is not able to interpret meaning of input parameters. In GRASS, input and output layer names (the real 'geodata') are delivered as simple strings. GRASS knows how to handle them, because they are already stored in GRASS's native data format. In a WPS environment, additional knowledge of data acquisition (for example a request from a feature server and its data schema) is needed for each geodata layer. Therefore, the WPS specification allows the usage of complex input parameters consisting of additional information regarding to data schemas, fetch URLs, and mime types (see Figure 2). Hence, the filter has to decide which of the simple GRASS parameters need to be further described and be represented by a complex input parameter and how. The distinction is easy when every geodata input layer is called 'input' and every output layer parameter 'output' (like in most cases in GRASS). However, an intersection operation implies two input layers, which cannot both be named 'input'. So, the XSLT filter is in trouble. So how to name them for easy recognition? Again, human GIS expertise is needed for a correct modeling of input layers.

Nevertheless, an XSLT filter can deliver a good base for WPS interface descriptions. Only in some rare cases human intervention is needed to create a valid DescribeProcess document.

3 Stateful vs. Stateless

GIS can be stateful on the one hand and stateless on the other. Stateful meaning being provided with a workspace to save intermediate results and the possibility to interrupt work, like in a desktop GIS. OGC web services are stateless. No trace of the process is left on the server after the response was sent to the client.

Processing GIS functionalities is usually preceded by defining a working environment. The environment has to include metadata information of e. g. the spatial reference system and the concrete data to be processed. This is unproblematic while operating on a persistent workspace, where data and metadata is imported once and remains usable for an unlimited number of tasks, as in classic desktop GIS. Even a GIS enhanced by a WPS interface will only run with a defined environment. Due to its stateless nature, this environment has to be created on-the-fly for each request (and deleted afterwards).

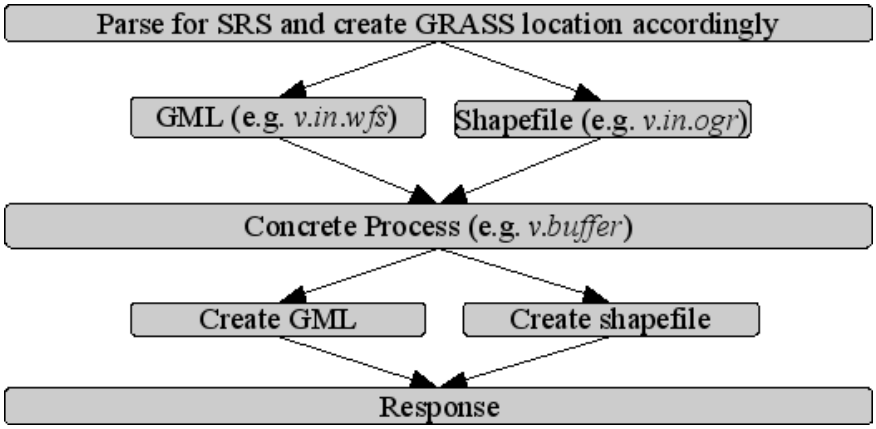

Figure 3: Sample workflow for v.buffer command.

GRASS or GIS in general do not define how to import and export data for a specific process (such as *v.buffer*). WPS does. The DescribeProcess document specifies which import and export data formats are supported. In the *Execute* process you have to include the data to be processed or provide a reference (like a WFS URL). Thus, for each *Execute* request the WPS has to decide dynamically, how to import and export data (which is not part of the underlying GRASS command). In the 52°North WPS implementation this is done by a dynamic plug-in architecture (see Figure 3).

```
<Create temporary GRASS location with desired SRS>

lynx -source "http://geoserver.itc.nl:8080/geoserver/wfs?[...]" >
     input.gml

v.in.ogr -o dsn=input.gml output=input          } GML-Import

v.buffer input=input buffer=0.01 output=output

v.out.ogr input=output dsn=output.gml format=GML   } GML-Export
```
Figure 4: Sample dynamically created GRASS script for v.buffer.

Figure 4 shows a sample script for the v.buffer command. It is created at runtime. For import and export regular GRASS commands are used (parsed in by the I/O plug-in mechanism), but they are invisible in the appropriate v.buffer DescribeProcess document.

4 Conclusion

Enhancing GRASS GIS by a Web Processing Interface is generally possible. This approach avoids a lot of reimplementation work of complex GIS functionalities, as they already exist in mature stand-alone GIS packages They can then be reused in a SOA respectively in spatial data infrastructures. New ways of GIS usage can now be realised, especially on machines with lower CPU power, such as mobile devices. Besides, occasional GIS users could choose from a variety of WPS processes instead of having to buy and install a full GIS package on their own.

The given approach could also be applied to other GIS packages, as long as they share similar properties. Further essential properties, problems and their solutions are described in my thesis.

The actual implementation for vector data shows that the applied approach results in a usable system. This and much more is as well presented in my upcoming thesis.

Two minor drawbacks need to undergo further examination: Some research has to be done regarding the semantic problems as described in Section 2. Furthermore, additional implementation of the various import and export data format filters is little more extensive than expected (Section 3).

Finally, the performance issue of sending e. g. large raster data sets over the Internet still remains. A smart caching solution or other concepts have to be found. A first GRID based WPS solution is developed by Baranski (2008). It leads to ideas of "bringing the service to the data" instead the other way round.

5 References

Baranski, B. (2008). 52° North WPS-G - A grid-enabled OGC Web Processing Service (WPS). Tech. rep., OGC-OGF Collaboration Workshop - The 22nd Open Grid Forum – OGF22.

Blake M. B., W. Cheung, M. C. Jaeger, A. Wombacher (2006). WSC-06: The Web Service Challenge. Proceedings of: The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06).

Burrough, P. A. and R. A. McDonnell (1998). Principles of Geographical Information Systems. Oxford University Press, New York.

Chen, X., W. Cai, S. J. Turner, Y. Wang (2006). SOAr-DSGrid: Service-Oriented Architecture for Distributed Simulation on the Grid. PADS '06: Proceedings of the 20th Workshop on Principles of Advanced and Distri-

buted Simulation (IEEE Computer Society, Washington, DC, USA), pp. 65-73.

Egenhofer, M. and R. Franzosa (1991). "Point-set topological spatial relations." International Journal of Geographical Information Systems 5(2): 161-174.

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

Neteler, M. and H. Mitasova (2007). Open Source GIS: A GRASS GIS Approach. Springer.

Open Geospatial Consortium Inc. (2007a). OGC Web Services Common Specification. Implementation specification. OGC 06-121r3.

Open Geospatial Consortium Inc. (2007b). OpenGIS Web Processing Service. Implementation specification. OGC 05-007r7.

Open GIS Consortium (2002). OpenGIS Abstract Specification Topic 12: OpenGIS Service Architecture. OGC 02-112.

Papazoglou, M. P. and W.-J. van den Heuvel (2007). "Service oriented architectures: approaches, technologies and research issues“. The VLDB Journal 16(3): 389-415.

Reed, C. (2004). Integrating Geospatial Standards and Standards Strategies into Business Process. An Open GIS Consortium (OGC) White Paper. Online available at: http://www.opengeospatial.org/pressroom/papers

Schäffer, B. (2007). Integrated Web Geoprocessing Workflow Composition and Deployment. Master’s thesis, Institute for Geoinformatics, University of Münster, Germany.