IMPLEMENTATION GRASS FUNCTIONALITY IN WPS SERVICE BASED ON .NET AND IIS

Adam Iwaniak[1], Bartosz Kopańczyk[2], Tomasz Kubik[3],
Pawel Netzel[4], Witold Paluszyński[3]

1. Wroclaw University of Environmental and Life Sciences, Institute of Geodesy and Geoinformatics, Grunwaldzka 53, 50-357 Wrocław, Poland
e-mail: adam.iwaniak@up.wroc.pl
2. GeoScope, Długa 58/11, 53-633 Wrocław, Poland
e-mail:kopanczyk@geoscope.pl
3. Wroclaw University of Technology, Institute of Computer Engineering, Control and Robotics, Wybrzeże Wyspiańskiego 27, 50-370 Wroclaw, Poland
e-mail: tomasz.kubik|witold.paluszynski@pwr.wroc.pl
4. University of Wrocław, Department of Climatology and Atmosphere Protection, Kosiby 8, 51-670 Wroclaw, Poland
e-mail: netzel@meteo.uni.wroc.pl

Abstract

GRASS (*Geographic Resources Analysis Support System*) is a widely used open source desktop GIS system. Its architecture is modular – each of its functions (or a group of functions) is implemented as separate program module running in the common GRASS environment.
WPS (*Web Processing Service*) is the web services specification developed by OGC (*Open Geospatial Consortium*) for sharing processing abilities over the network according to the SOA (*Service Oriented Architecture*) paradigm. The specification provides the definitions of the service interface, but does not describe any specific processing functionality offered by the service nor the way of its implementation. This part is left to the WPS service providers who can build their own processing algorithms behind the service interface and offer them to the service clients through the *Execute* operation. This opens an opportunity to incorporate the GRASS functionality in the service design; especially that modular architecture of GRASS is highly suitable for this task.
The number of WPS server implementations with a GRASS as a processing backend is rather small. One of the examples is PyWPS. This paper describes another implementation of WPS (an extendable software framework) with GRASS modules used as processing engines.
The implementation described conforms to the OGC WPS version 1.0 specification. The language of implementation is C# language. The IIS (*Internet Information Server*) and .NET framework are technologies this implementation is based on. The system works in the Windows environment. The connectivity platform between WPS service

and GRASS modules is scripting system. The prototype installation offers only few spatial analyses, like: shadowing, morphometric parameters calculation, raster calculator etc. However, it is very easy to extend it with new functions.

1. Introduction

OGC Web Services technology is a specific and GIS-oriented implementation of a more general concept of the SOA technology. Web Services technology has an ability for remote communication with other services and components, their mutual identification and usage of functionality, similar to RPC (*Remote Procedure Calling*).
One of the web services specifications with a great potential, but not fully utilized yet in the building of SDI (*Spatial Data Infrastructure*), is Web Processing Service – a standard released by the OGC (OGC-standard; Keens, 2006). This standard provides the description of the service interface that may serve as a façade to the interfaces of other services already defined or just planned, but without detailed specification provided. More over, WPS entities are well suited for process chaining – the strategy used in GIS analysis, where the final objective is achieved as an outcome of series of processing steps (Friis-Christensen et al., 2007; Kubik et al. 2007).
The recent version of the standard (1.0.0) appeared on 8 June 2007. Since then a few implementations of WPS frameworks were released. These frameworks are software solutions that offer generic functionality of the service, which has to be completed with a custom code to provide specific functionality. Among these frameworks, only a few offer WPS server implementations with GRASS as a backend for processing. The example is PyWPS – the WPS implementation by 52°North described by Brauner and Schaeffer (2008) or implementation described by Cepicky and Becchi (2007). In general, WPS can serve as a gate leading to any kind of processing (Michaelis and Ames, 2009).
The paper concern WPS as a web service merged with GRASS, offering GRASS functionalities over the network. The implementation of WPS described derives from Internet Information Server and the .NET technologies. The system works under the Windows environment. The WPS server development platform is C# language. The implementation conforms to the OGC WPS version 1.0.0 specification. A scripting system provides connectivity between WPS server and GRASS modules.

2. GRASS

GRASS (*Geographic Resources Analysis Support System*) is a desktop, modular GIS system provided under the GNU GPL license. The system has a long history. At the beginning, GRASS was a military project. It was a raster-oriented GIS system. Subsequently, GRASS was published as an open source project. A community of users and developers evolved around the new free system. Now, GRASS is one of the leading free and open source desktop GIS systems. It supports all kinds of GIS analyses: raster, vector, image etc. Additionally, there are many ready to use algorithms focused on

particular problems: radiation calculation, flood modeling, etc. (Neteler, Mitasova, 2008).
GRASS is based on the system command line interpreter. So, it is possible to use GRASS commands in other GIS systems/services by preparing textual scripts and building data processing chains. On the other hand, GRASS is an open source project and it is possible to write chains of processing algorithms directly in the programming language (GRASS-manual).
GRASS uses the GDAL/OGR library as a basis for data exchange. Data formats provided by GDAL can be read into and exported from the GRASS system. Apart from the GDAL library, GRASS has a native implementation of import/export procedures developed for individual data formats. It is very convenient to read input data into GRASS data structures and perform advanced data processing chains because the internal format is managed in the fastest way.
Each GRASS set of data, or set of GIS data layers, can be stored in a separate structure, called 'location', which defines the extent, resolution and geographical projection of data. Individual users can create separate substructures – datasets. They can manage access rights to data stored in datasets. Because creation and deletion of locations and datasets can be done automatically, all externally called processing can be performed in some separate, temporary data spaces.

3. WPS implementation

The implementation of the OGC WPS described here (plWPS) uses the MS IIS (*Microsoft Internet Information Server*). All messages and requests are forwarded from the IIS Server to the *RequestProcessor* class. Firstly, an instance of this class creates a description part of the process. Next, this description and additional parameters from the request are transmitted to the execution part of the system – the *AssemblyController* class. All processes' implementations are contained in a dedicated directory as dynamically linked libraries (DLL files). In the case of GRASS procedures, the *AssemblyController* gets an instance of *GrassManager*, which encapsulates the GRASS native technology. If all necessary parameters get validated successfully, then the *GrassManager* runs a shell script starting a GRASS task. *GrassManager* also monitors the state of this task. Upon the completion of this process, the client receives a response with information about the status of the task just ended. A WPS process can run synchronously or asynchronously. In the synchronous case, the response contains the results of processing. In the asynchronous case, the client receives details on where to look for the results of the processing. The schema in the figure 1 represents the building blocks of WPS implementation described here.

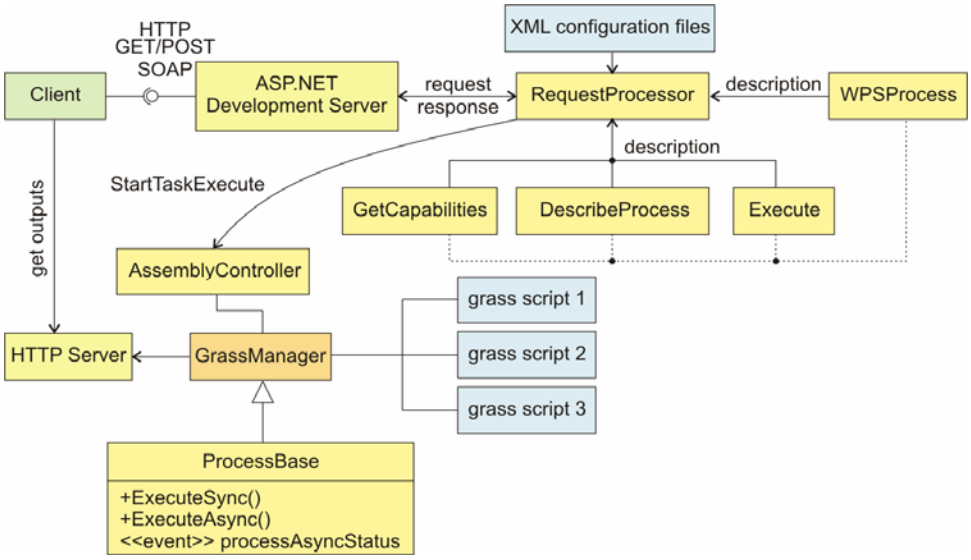

Figure 1. plWPS building blocks

4. Building processing algorithms based on GRASS

GRASS modules work in a common environment. The environment provides the parameters, like spatial database location, data location, the mapset, GRASS binaries location etc. To start the GRASS module it is necessary to setup the environment properly. It can be done by setting up shell variables and prepare appropriate configuration files. Both setups can be prepared on the fly by a shell script. Under the Linux/Unix systems there is no problem to make this work. The Windows GRASS installation package also provides a shell scripting engine.

Another problem to solve is the data storage. The obvious solution is to store all data in a native GRASS format. This solution gives the best performance. GRASS uses GDAL/OGR and provides a few additional native procedures to import and export spatial data.

Each WPS process should operate on its own, separate data set. The data set is defined by projection, extent, and resolution. The GRASS system uses data organized in locations and mapsets. The logical structure of data is implemented in the structure of directories in the filesystem. So, for each starting WPS process the WPS server creates a new folder for a new location, sets projection and region of interest, and creates a default mapset. The data sent by the WPS client are imported and stored in the newly created location. The name of the location is built using process id. After serving a client's request, the WPS server removes the whole location structure recursively. The following diagram illustrates a WPS server and a GRASS system activities during serving a synchronous client request:

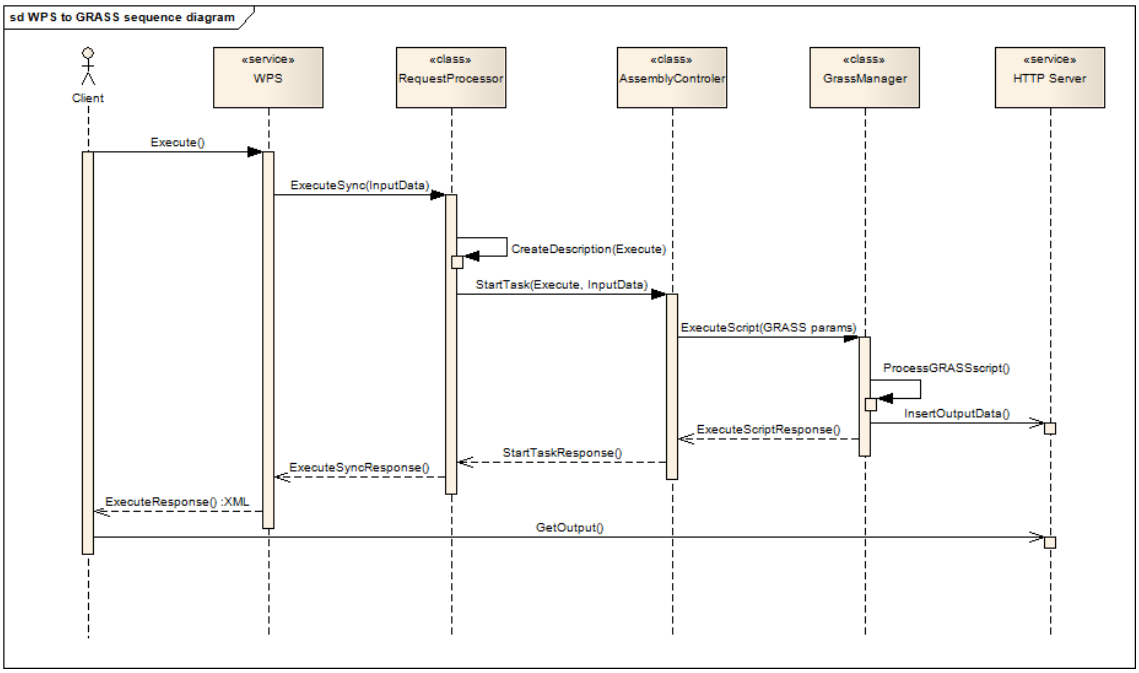

Figure 2. UML sequence diagram of plWPS system activities during serving a synchronous client request

5. An example of a WPS function

Let us look at the WPS/GRASS implementation of a morphometric features calculation. A GeoTiff file containing an orthophoto map is an input data sent to the service by a client. The DTM is stored in the GeoTiff Float32 data format. The DTM is shown in the figure 3.

Figure 3. Sample input data

First of all, the processing methods served by the WPS server can be discovered trough the request for service's capabilities as in the following example of the request encoded in XML:

```
<wps:GetCapabilities xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ows="http://www.opengis.net/ows/1.1"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
../wpsGetCapabilities_request.xsd" language="pl-PL" service="WPS">
        <wps:AcceptVersions>
                <ows:Version>1.0.0</ows:Version>
        </wps:AcceptVersions>
</wps:GetCapabilities>
```

The response from the service will contain all required information about the service, encoded in XML as in the following example:

```
<?xml version="1.0" encoding="utf-8"?>
<wps:Capabilities xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ows="http://www.opengis.net/ows/1.1" service="WPS" xml:lang="en-GB"
xmlns:wps="http://www.opengis.net/wps/1.0.0">
  <ows:ServiceIdentification>
    <ows:Title xml:lang="en-GB">WPS service </ows:Title>
    <ows:Abstract xml:lang="en-GB">Experimental WPS service</ows:Abstract>
    <ows:Keywords>
      <ows:Keyword xml:lang="en-GB">geoprocessing</ows:Keyword>
    </ows:Keywords>
    <ows:ServiceType codeSpace="cs">WPS</ows:ServiceType>
    <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
    <ows:Fees>NONE</ows:Fees>
    <ows:AccessConstraints>NONE</ows:AccessConstraints>
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
                <!--Omitted section about service provider-->
  </ows:ServiceProvider>
  <ows:OperationsMetadata>
<!--Omitted section about provided WPS operations. Possible are: GetCapabilities,
DescribeProcess, Execute-->
  </ows:OperationsMetadata>
  <wps:ProcessOfferings>
    <wps:Process processVersion="1.0">
      <ows:Identifier codeSpace="">ReliefShading</ows:Identifier>
      <ows:Title xml:lang="en-GB">Relief shading</ows:Title>
```

```
    <ows:Abstract xml:lang="en-GB">Process does the relief shading. Geotiff file is the source of shading, PNG file comes as the process result.</ows:Abstract>
    <ows:Metadata type="simple" xlink:title="geotiff" />
    <ows:Metadata type="simple" xlink:title="png" />
   </wps:Process>
  </wps:ProcessOfferings>
  <wps:Languages>
   <wps:Default>
    <ows:Language>pl-PL</ows:Language>
   </wps:Default>
   <wps:Supported>
    <ows:Language>en-GB</ows:Language>
   </wps:Supported>
  </wps:Languages>
  <wps:WSDL href="http://wps.bqsystems.com.pl:8081/wps.wsdl" />
</wps:Capabilities>
```

The *ProcessOfferings* section contains the description of all processes offered in the service. In the example above this section contains only one description – *ReliefShading*, which is associated with the previously mentioned GRASS morphometric feature calculation. Now, the request for processing can be prepared in the form of WPS *Execute* request, as in the following example:

```
<wps:Execute service="WPS" version="1.0.0" language="pl-PL"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0 ../wpsExecute_request.xsd">
        <ows:Identifier>ReliefShading</ows:Identifier>
        <wps:DataInputs>
                <wps:Input>
                        <ows:Identifier>geotiff_source</ows:Identifier>
                        <ows:Title>Input geotiff file</ows:Title>
                        <ows:Abstract>URL to the input geotiff file</ows:Abstract>
                        <wps:Reference
xlink:href="http://www.bqsystems.com.pl/wps/temp.tif"/>
                </wps:Input>
        </wps:DataInputs>
        <wps:ResponseForm>
                <wps:ResponseDocument storeExecuteResponse="false" status="true">
                        <wps:Output asReference="true" mimeType="image/png"
encoding="UTF-8">
```

```
                    <ows:Identifier>png_output</ows:Identifier>
                    <ows:Title>PNG file outpu after processing</ows:Title>
                </wps:Output>
            </wps:ResponseDocument>
        </wps:ResponseForm>
</wps:Execute>
```

The referenced URL in the example is a place, from where the service can obtain an input file (download it) for processing. Each GRASS process is coded as a shell script file. Following is the script for relief shading:

```
#!/bin/sh
TASKID=$$
INPUTFILE=$1
OUTPUTFILE=$2

GRASSDIR=/c/GRASS
TEMPDIR=/c/GRASS/tmp
TEMPDIRWIN=c:\\grass\\tmp
SKELLOCATION=/c/dane/wps/skel
DATADIR=$TEMPDIR/$TASKID
GISRC=$DATADIR/.grassrc6
LOCATION=WPS
MAPSET=PERMANENT
mkdir $DATADIR
mkdir $DATADIR/$LOCATION
mkdir $DATADIR/$LOCATION/$MAPSET
#mkdir $DATADIR/$LOCATION/$MAPSET/.tmp
echo "src:$SKELLOCATION/* dst: $DATADIR/$LOCATION/$MAPSET"
cp $SKELLOCATION/* $DATADIR/$LOCATION/$MAPSET
echo "GISDBASE: $TEMPDIRWIN\\$TASKID
LOCATION_NAME: $LOCATION
MAPSET: $MAPSET
" > $GISRC

export GISBASE=$GRASSDIR
export GISRC
export PATH=$GRASSDIR/bin:$GRASSDIR/extrabin:$GRASSDIR/scripts:$GRASSDI
R/lib:$GRASSDIR/extralib:$GRASSDIR/sqlite/bin:$PATH
export LD_LIBRARY_PATH=$GRASSDIR/lib:$GRASSDIR/extralib:$GRASSDIR/sqlite
/lib:$$LD_LIBRARY_PATH
export GRASS_PNGFILE=$OUTPUTFILE
export GRASS_TRUECOLOR=TRUE
export GRASS_WIDTH=900
```

```
export GRASS_PNG_COMPRESSION=1
export GIS_LOCK=$$

g.proj -c georef=$INPUTFILE

r.in.gdal input=$INPUTFILE output=w1

if [ -f $DATADIR/$LOCATION/$MAPSET/cell/w1 ]
then
        g.rename rast=w1,w1c
else
        r.composite red=w1.red green=w1.green blue=w1.blue output=w1c
fi
r.shaded.relief map=w1c shadedmap=w2
r.out.png input=w2 output=$OUTPUTFILE
rm -fr $DATADIR
```

The script code is divided into four segments: setting up a GRASS session and spatial location in the GRASS database, importing input data, processing, exporting results and freeing data storage in the GRASS database.
After finishing the work the server provides an URL pointing to the location of the results. The user can download the results then – in this case a PNG file. The PNG format can be previewed in a web browser window. So, a user can copy the obtained URL to the URL input field of a browser to see the results. The result of this processing is shown in the figure 4.

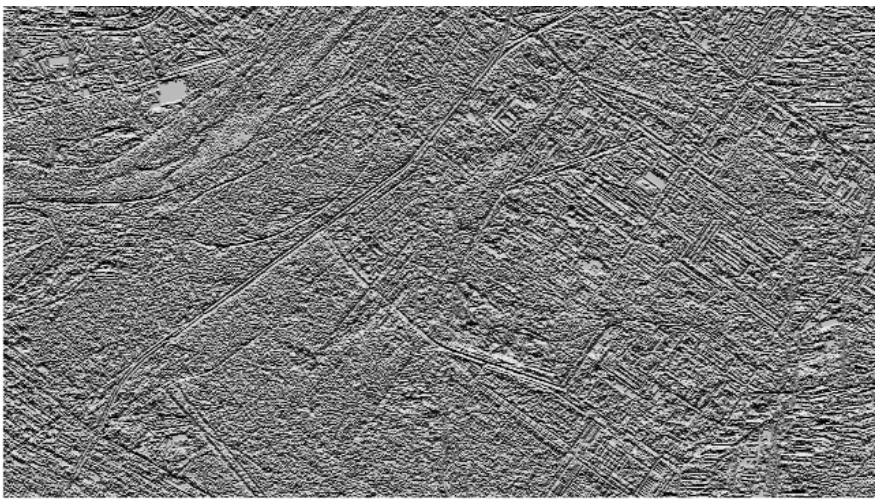

Figure 4. A result of plWPS processing

6. Summary

The WPS standard allows GIS processing abilities sharing over the network. A user equipped with a thin client can send (or point to) source data to the server and obtain the results of processing. The WPS implementation, incorporating GRASS as a processing backend and set of batch scripts, is easy to extend. By adding new scripts any algorithm built into GRASS can be exposed to the clients trough the network. It is possible to implement processing chains dedicated to the particular branches of science like hydrology, meteorology, geology.

Acknowledgements

This work has been sponsored by the Polish government scientific funding during the years 2007-2010 as a research and development project.

References

Brauner J., Schaeffer B., 2008. Integration of GRASS Functionality in Web based SDI Service Chains, *OSGeo Conference Proceedings Track*, Cape Town, South Afrika.

Cepicky. J. Becchi. L.. 2007. Geospatial Processing via Internet on Remote Servers - PyWPS. *OSGeo Journal*, Vol. 1. Available at http://www.osgeo.org/journal/volumel

Friis-Christensen A., Lutz. M., Ostlinder. N., Bernard. L., 2007. Designing Service Architectures for Distributed Geoprocessing: Challenges and Future Directions. *Transactions in GIS*, 11(6), pp.799-818.

GRASS-manual, *GRASS Programmer's Manual*, internet source, http://download.osgeo.org/grass/grass7_progman/

Keens S., 2006. OWS-4 WPS IPR: Discussion, findings, and use of WPS in OWS-4, Technical Report, Open Geospatial Consortium, Wayland, Document Number OGC 06-182

Kubik T., Paluszyński W., Kopańczyk B., Iwaniak A, 2007, Implementing a WPS service for automatic feature recognition in orthophoto maps. In: *Proceedings of XXIII International Cartographic Conference*, Moscow, Russia, 4-10 August 2007.

Michaelis, C.D.; Ames, D.P., 2009. Evaluation and Implementation of the OGC Web Processing Service for Use in Client-Side GIS. *GeoInformatica*, 13(1), pp.109-120.

Neteler M., Mitasova H., 2008. Open Source GIS: A GRASS GIS Approach. Third Edition. *The International Series in Engineering and Computer Science*: Volume 773, Springer, New York

OGC-standard, *OGC Web Processing Service Interface Standard*, internet source, http://www.opengeospatial.org/standards/wps