OGC WEB PROCESSING SERVICE AND IT'S USAGE

Jáchym Čepický[1]

[1]Help Service – Remote Sensing, Černoleská 1600,
256 01, Benešov u Prahy, Česká republika
jachym@bnhelp.cz

Abstrakt. Zatímco zkratky jako WMS, WFS nebo WCS ji zdomácněly v našem geoinformačním slovníku, WPS je zatím stále na pokraji zájmu. OGC Web Processing Service (WPS) postupně dospívá od verze 0.4.0 k verzi 1.0.0. Je to standard, který definuje způsob, jakým jsou funkce GIS distribuovány v prostředí počítačových sítí. Funkce GIS jsou ve standardu popisovány jako "procesy" a mohou být od těch jednoduchých (např. překryv dvou vektorových vrstev) až po velmi komplikované (např. globální modely změn klimatu). WPS umožňuje, aby geografické výpočty, které jsou náročné jednak na hardware, ale často i na implementaci, byly spouštěny na vzdáleném severu a odlehčily tak zatížení pracovní stanice.

V současné době je nemnoho implementací této specifikace, jak na stran serveru, tak na stran klienta. Tento příspěvek představuje několik aplikací, které jsou vytvořeny pomocí tohoto standardu. Na straně serveru je většinou využit program PyWPS, který nedávno dospěl do verze 2.0.0 (implementuje standard WPS 0.4.0).

Klientské aplikace mohou být tvořeny na míru konkrétním procesům na konkrétních serverech a nebo obecně tak, aby byly pokud mono schopny komunikovat s libovolnými servery a jejich procesy. Příkladem prvního aplikace je "WPS demo", demonstrující využití OGC WPS v prostředí webového prohlížeče nebo "Prefarm" -- portál sloužící k výpočtu optimálních dávek hnojení. Tito klienti tak říkajíc "znají své procesy" a neumí komunikovat s libovolnou sadou nástrojů z cizích (jim neznámých) serverů. Druhou skupinu reprezentuje WPS modul mapového klienta OpenLayers a WPS zásuvný modul proprogramy uDig a Open Jump. Tyto moduly jsou vytvářeny tak, aby se dokázaly "dohodnout" s jakýmkoliv serverem. Povyšují funkce prohlížeček geodat naplnohodnotný GIS.

Klíčová slova: OGC, WPS, GIS, PyWPS, GRASS, Processing Service

Abstract. Where shortcuts as WMS, WFS or WCS are well known in our geoinformation vocabulary, WPS is still rather new. OGC Web Processing Service (WPS) is approaching from version 0.4.0 to version 1.0.0. A WPS can be configured to offer any sort of GIS functionality to clients across a network, including access to pre-programmed calculations and/or computation models that operate on spatially referenced data. A WPS may offer calculations as simple as subtracting one set of spatially referenced numbers from another (e.g., determining the difference in influenza cases between two different seasons), or as complicated as a global climate change model. This processes can be calculated on remote servers and so, the local stations can be common desktop computers.

There are not so many implementations of this standard at both, server and client side. In this paper, several applications are presented. On the server side, PyWPS is mostly used. PyWPS is now in it's version 2.0.0 which implements OGC WPS 0.4.0.

Two approaches can be taken, while developing the WPS clients. Clients can be constructed, so that the processes and servers are hard coded in the source code. The client "knows it's processes" and is not able to communicate with other servers. As example, "WPS demo" or "Prefarm" can be taken. WPS demo demonstrates usage of WPS in the web browser. Prefarm is application designed for farmers, which can calculate optimal amount of fertilizer over fields. Second approach is represented by WPS plug-in for uDig and Open Jump, as well as WPS module for Mapclient OpenLayers. This programs are designed, so that then can communicate with any server. They are upgrading general geodata viewer to full featured GIS.

Keywords: OGC, WPS, GIS, PyWPS, GRASS, Processing Service

1 Introduction

The Open Geospatial Consortium, Inc.® (OGC)[1] is a non-profit, international, voluntary consensus standards organization that is leading the development of standards for geospatial and location based services. OGC specifications are technical documents that detail interfaces or encodings. Software developers use these documents to build support for the interfaces or encodings into their products and services. One of this document OGC Web Processing Service (WPS). It is relatively new standard (compared to other, more used standards, like OGC Web Mapping Service (WMS) or Web Feature Service (WFS)). The document number 05-007r4 describes version 0.4.0 of the WPS standard. Request for comments to this standard was published February 2006.

The standard describes the way, how geospatial operations (referred as "processes") are distributed across networks. Now days, there are only two implementation of the standard: 52north WPS and PyWPS. Also client applications, able to communicate with the server via WPS standard are only few. Interesting is, that also big projects and companies did ignore this standard till now. Several running applications will be presented in this paper, which are currently available [1].

2 OGC Web Processing Service

In the document number 05-007r4 communication between server and client is described. The communications consists of three types of request-response pairs. Request can be in Key-Value-Pairs encoding (KVP) or it can be send to the server as XML document. Server response is always formated as XML document. Similar to other OGC specifications, the requests are GetCapabilities, DescribeProcess and Execute.

2.1 GetCapabilities

As response to the GetCapabilities client request, Capabilities XML document is returned. It consists of two main parts: ServiceIdentification and ProcessOfferings. The structure of

[1] Open Geospatial Consortium http://opengeospatial.org

ServiceIdentification part is mainly documented in OpenGIS® Web Service Common Implementation Specification [12]. Server provider as well as constrains, fees and other additional data (if any) are listed here. In the ProcessOfferings part, list of on server available processes is appended. Client, which is able to parse the Capabilities document correctly, has list of processes offered by the server. After the document is parsed, it can request more detailed process description of selected processes.

2.2 DescribeProcess

At DescribeProcess request, ProcessDescription XML document will be returned. It contains detailed process characteristics, as it's title, identifier and abstract with more detailed informations. Also necessary inputs and their types are listed here. Results of geospatial calculations will be stored in Outputs section. Client, which is able to parse XML ProcessDescription document, gets general overview, which inputs it has to request from the user in order to be able handle them to the server and let the process to be executed. It can also prepare it self at the server response, because process outputs, their formats and types are described in the second part of the document as well.

2.3 Execute

At Execute request, client sends the input data and/or values to the server, waiting for the server response.After the final response with process outputs is obtained, client can display the data to the user.

2.4 Data input and output types

In- and Output data can be of three different kinds:

- **LiteralData** – character strings, integer numbers as well as double numbers
- **ComplexValue** and **ComplexValueReference** – those are raster, vector or other (large) data files – maps in various formats.
- **BoundingBox** – two pairs of coordinates.

ComplexValue differes from ComplexValueReference in the fact, that the data are part of the request/response, where ComplexValueReference does handover only URL reference to the location, where the data can be downloaded.

2.5 WPS Data Management models

Different approaches can be taken, to pass input data to the server. They can be combined together, as well as only one approach can be selected.

For Input data:

- **Data are stored on the server.** So it looks at least, from the client point of view. No (spatial) input data are required to be send with the request to the server. The data can be stored at the server directly, as well as at remote servers and WPS server can request them via SQL, WCS or other standards. Client does not know anything about this process.
  Example of the design can be seen in any Shortest path calculation: All necessary input data (road network) is usually stored at the server, clients overhands only start and stop coordinates.

- **Data are shipped as part of the request documented.** In this case, request is usually send as XML document. Vector data, can be stored in several XML formats, namely GML, SVG, GeoRSS and others. This XML documents can be included into XML request as its part. For raster data, CDATA section can be used, in which any text or binary data can be stored.
  For example, buffer process requires input vector object, so it can make buffer around it.
- **Only references are part of the request document**, server will have to download them from remote server. This enables us, to make the client as simple as possible: No data do have to be stored at client side. Client can only point to one service to another (WPS to WCS or WFS in this case).
  For example, image classification process can get URL to Web Coveradge Service, where it gets necessary input raster maps.
  This last example demonstrates in our opinion the power of web services and their possible usage. Web services are bind together to provide working solution.

Above mentioned list implies also for data outputs. Output XML file can contain directly vector or raster maps, embed as included XML or as content of the CDATA object. It is also possible, to send only references to output data, which can be later downloaded. Finally, server can return raw data directly, as response to the WPS Execute request.

2.6 Asynchronous vs. Synchronous process execution

Some geospatial calculations can take longer time, in therms of hours, days or even weeks. This can easy lead to Server Timeout error and also, it is hardly possible to get more informations about the calculation progress. Therefore "status" input parameter can be used. If set to "true", then (if supported) the server will send Execute response back immediately after it obtains client's request with a message, that the request was accepted together with an URL, where the client can regularly check calculation progress and will find out what is happening as well as guessed percentage of remaining time.

2.7 Issues of OGC WPS 0.4.0 standard

Christopher Michael and Daniel P. Ames described in their paper [2] six proposals to WPS 0.4.0 standard, which are trying to fix some missing features of the format. Namely

- Adding section, which informs client application, how the request input data from the user. Currently, only data types are supported (String, Complex data). But e.g. number format (integer or double format), raster or vector data selection is missing.
  New PromtMethod element is proposed, which would include defined type of user input, such as "browseforraster", "browseforvector" or "getboundbingbox".
- Adding section of data available at the server, so the client could make data selection by itself.
- Returning raw data as Execute response should be avoided – XML document should be always returned back.
- It is not possible to cancel running process. Methods for calculation break should be added as well.
- Each process should have single URL – currently, all processes do have common URL. This enhancement would make the WPS standard inconsistent, if compared to other OGC standards.

- More highly structured exception system should be implemented. Current system is only primitive and offers Exception codes only for limited number of cases (ServerBusy, FileSizeExceeded, InvalidParameterValue, NoAplicableCode).

Also other improvements, such as missing internationalization support, could be implemented in new standard versions. Some of them were included in to OGC WPS 1.0.0, which was published in the summer 2007.

2.8 OGC WPS 1.0.0

New version of the standard [3] introduces new features, which are enlarging all possible ways of WPS usage, as well as making life of programmers easier, namely:

- **SOAP and WSDL** usage. SOAP (Service Oriented Architecture Protocol) is a protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS. SOAP forms the foundation layer of the Web services stack, providing a basic messaging framework upon which abstract layers can be built [5].
  The Web Services Description Language (WSDL) is an XML-based language that provides a model for describing Web services. The WSDL defines services as collections of network endpoints, or ports. WSDL specification provides an XML format for documents for this purpose. It describes the public interface to the web service.
  WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special data types used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL [6].
- **New KVP request econding definition.** In WPS 0.4.0, input data definition is rather primitive. All input data are stored as key-value pairs, separated with comma. For example, part of the URL would be formated as "...&DataInputs=key1,value1,key2,value2&...".
  New input KVP encoding has more complicated structure. Each input is separated with semicolon (;). Input parameters are separated by at (@) mark. Example of such input formating could like like follows: "...&DataInputs=key1=value1;key2=@xlink:href=http%3A%2F%2Ffoo.bar&...".
- **More concrete input description** in DescribeProcess document. New input attributes are defined not only by type, but several additional attributes can be added, including maximal file size.
- Support for **internationalization** of the server. Requested language can be set with help of "lang" parameter.

We can see, that some of the proposals described in [2] were implemented, however, most not. For example, in our opinion, acceptance of the PromtMethod (or similar) tag, would be highly welcomed by GUI applications developers. Also, no security issue is not taken in account. Usually, the open to the wide Internet community.

2.9 WPS implementations

Before we start to talk about programs implementing WPS standard, we have to mention web-based geoprocessing systems and approaches similar to WPS, that have been implemented by various entities. Most notably, the Environmental Systems Research Institute (ESRI) product ArcInfo 8.3 (ESRI, 2003). It contains a feature called the Geoprocessing Server, which ran on large-scale UNIX servers to perform geoprocessing on behalf of ESRI client software which submitted jobs for processing. The ESRI Geoprocessing Server protocol is proprietary and closed such that only ESRI software is able to make use of the remote processing capabilities. Interestingly, this feature was removed from the following version (ArcGIS 9.0). A similar but subtly different feature was introduced in ArcGIS Server 9.2. However, unlike WPS, the ESRI implementation is not compatible with non-ESRI products and a closed, proprietary communications protocol preventing it from being adopted at large or studied in a non-ESRI environment.

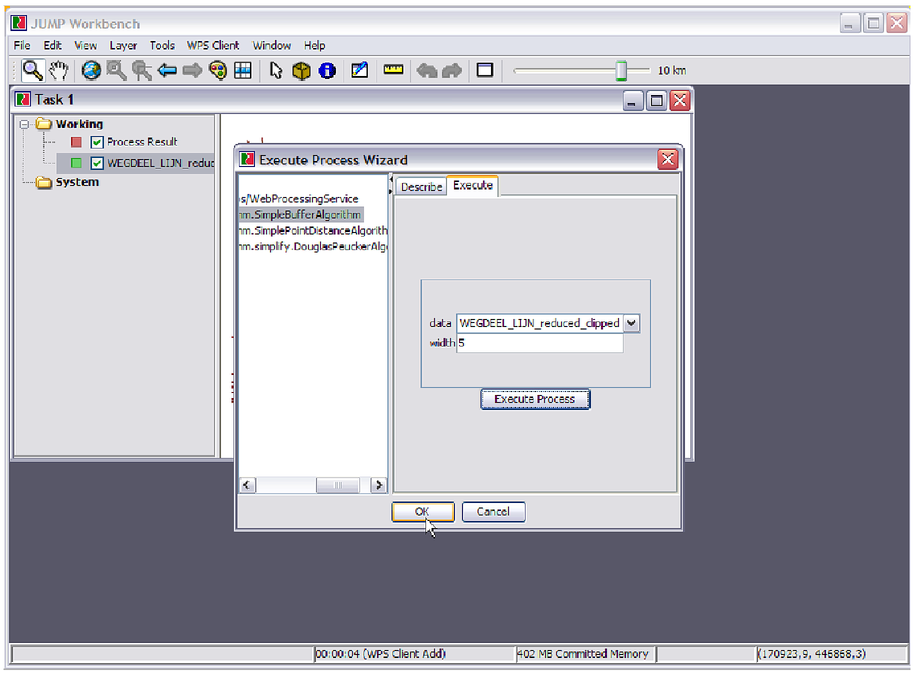


Fig. 1: *WPS plug-in for OpenJump GIS viewer.*

Currently (autumn 2007), only several implementations of OGC WPS are know [4] and all from them are implementing the 0.4.0 version of the standard. However it is known, that 52north WPS server development team is actively working of 1.0.0 WPS standard implementation.

Currently known WPS server projects:

- **Deegree framework** is Java environment, which implements most of OGC standards. It is used mostly in german state administration[2].

[2] deegree: http://deegree.org

- **WPSint** a JAVA plug-in for Spring (full-stack Java/JEE application framework). Although it implements 0.4.0 version of the standard, it can be used via WSDL/SOAP interface[3].

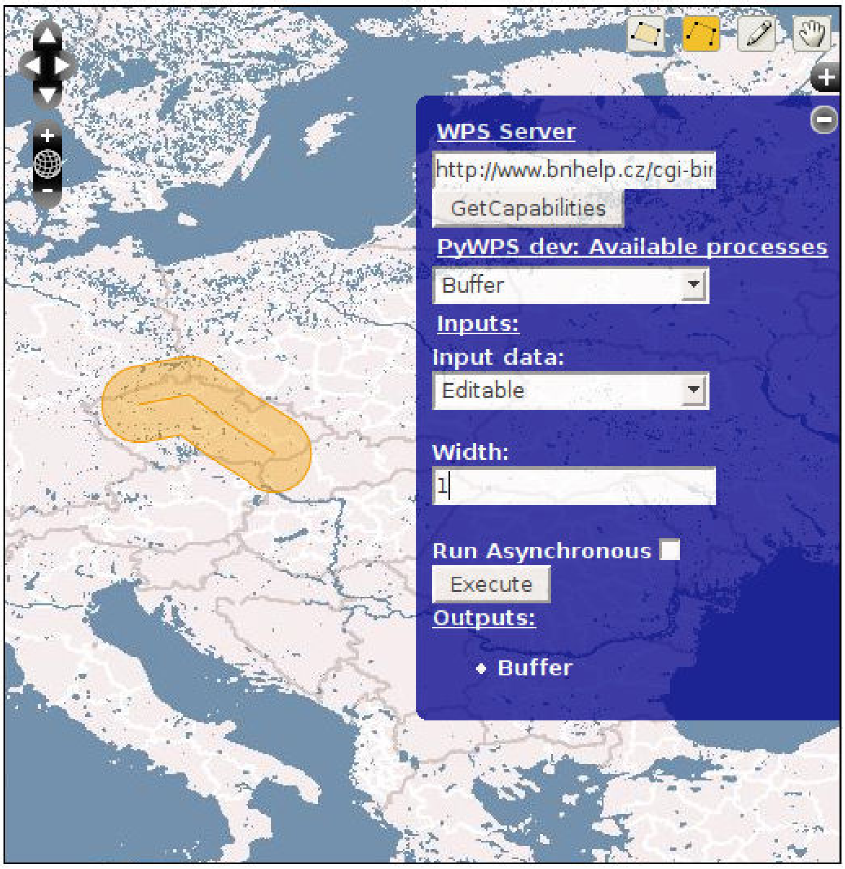


Fig. 2: *OpenLayers WPS plugin. The vector line was digitized using standard OpenLayers tool. The line was send as GML file to Buffer process installed in PyWPS. Returned GML file with buffer was later displaied in OpanLayers as new layer.*

- **PyWPS** states for *Python Web Processing Service*. It is written with direct support for GRASS GIS and other GIS command-line tools. More about this program later[4].
- **52 North WPS** is written in Java, as plug-in for Java Tomcat serverlet container. Part of 52 North WPS are also plugin-client applications, for uDig and Open-Jump GIS geodata viewers[5].
- **GeoConnections WPS** is implementation of one of the older versions of the standard (0.2.0). The project seems to be no longer supported[6].

Even though to code WPS client is notably easier, than to write WPS server. Only few projects can be found in todays repositories. Namely

[3] WPSint: http://wpsint.tigris.org
[4] PyWPS: http://pywps.wald.intevation.org
[5] 52North WPS: http://52north.org/index.php?option=com_projects&task=showProject&id=21&Itemid=127
[6] GeoConnections WPS: http://www.geoconnections.org/en/communities/developers/standards/fa=technical.webprocessing_service

- **52 North WPS plugin for uDig and Open-Jump**. Both are written in Java and can be used with current versions of named programs[7].
- **OpenLayers WPS Control** class is new OpenLayer.Control class. OpenLayers is geodata viewer, written in JavaScript, used for displaying maps in web browsers. This WPS plugin makes it possible, running geospatial operations using general web browser[8].

Several other projects are working with WPS, but the clients are proprietary – designed for work with specified server and specified processes. Among others, WPS Demo [7] can be named.

PyWPS is project, which is developed since 2006, and tries to implement OGC WPS standard in it's 0.4.0 version. It is written in Python programming language. The main goal of PyWPS is, that it has been written from the beginning, with directo support for GRASS GIS. So, PyWPS can be understand, as kind of translation library, which translates requests complain to WPS standard, overhands them to GRASS GIS [9] or other command line tool (such as GDAL/OGR, PROJ.4 or R statistical package), monitors the calculation progress and informs the user and after the calculation is completed, it returns back it's result.

PyWPS released under terms of GNU/GPL licence. Currently, version 2.0.0 is available. It is actively maintained by Help Service - Remote Sensing company as one of our projects. All applications described in this paper are using PyWPS as back-end.
Currently, we are continuing with PyWPS development and would like to introduce implementation of WPS 1.0.0 standard by Summer 2008 [8].

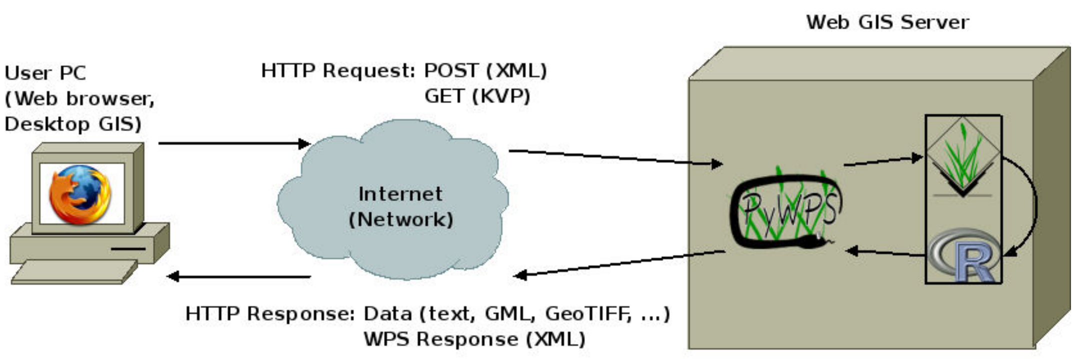


***Fig. 3:** Schema of PyWPS design.*

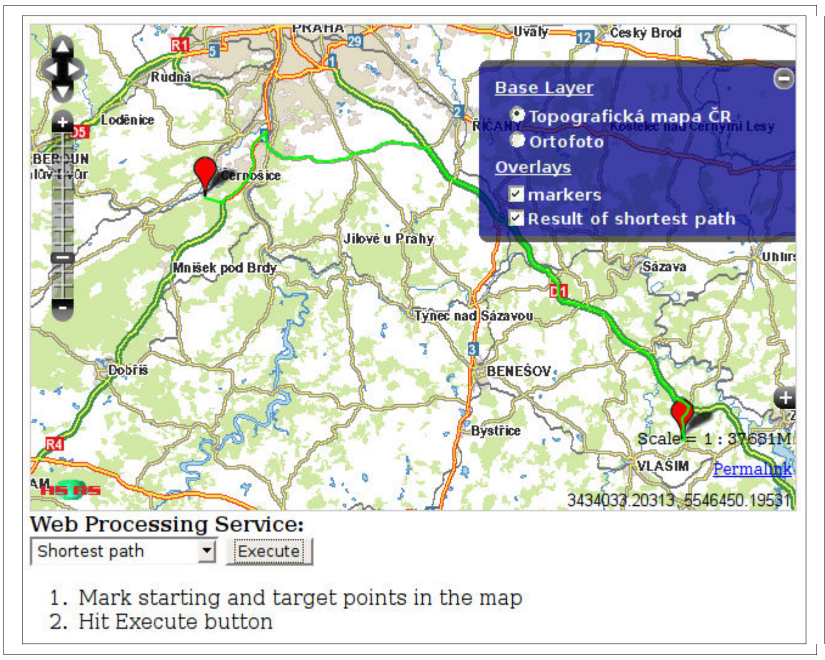
3 Usage of OGC (Py)WPS

Several running examples, which are demonstrating WPS usage can be visited on the Internet. Other examples are running as intranet applications. We would like to give you a short introduction to some of them.

[7] uDig WPS client: http://incubator.52north.org/twiki/bin/view/Processing/52nUdigWPSClient
[8] OpenLayers WPS Control: http://dev.openlayers.org/sandbox/jachym/openlayers/examples/wps.html

3.1 WPS Demo

From summer 2006, WPS Demo application can be used, which was developed by Help Service Remote Sensing company [7]. The demo is continuously updated, so it can demonstrate all WPS possibilities. Todays version uses PyWPS 2.0.0 as back-end and OpenLayers map client as front-end for displaying the data. The application is entirely written using JavaScript language. All processes are using GRASS GIS as background geoprocessing tool and GDAL/OGR and PROJ.4 libraries, for data transformation.



Web Processing Service:
Shortest path
Execute

1. Mark starting and target points in the map
2. Hit Execute button

Fig. 4: *Shortest path in new WPS Demo. OpenLayers are used for map display.*

Currently available processes are:

- **Shortest path** calculation, which uses v.net.path module of GRASS GIS. The output vector file is exported as GML vector format which is embed into Execute Response document. OpenLayers client than parses output document, converts the GML file onto SVG format and displays it directly in the browser window.
- **Unsupervised image classification** is another process. The user has to zoom to specified region and then she can process the classification by setting desired number of classes. One of the input parameters is WCS URL pointing to Landsat imagery file. When the file is downloaded and imported, unsupervised classification with help of i.cluster and i.class is performed.
  Resulting raster file is converted to PNG file and returned back to the client together with reference to original GeoTIFF file. PNG file can be displayed directly as one of the layers in OpenLayers Map object.
- **Visibility analysis** works similar to previous process. The user have to define coordinates of the viewport (per mouse click) and after a while, resulting PNG file from r.los is displayed.

- **Flow analysis** calculates raster map of flowlines, using r.flow module. User have to zoom to desired region. The flow analysis is done on the digital elevation model, stored on the server. Resulting GeoTIFF or PNG files can be then displayed by the client.

3.2 Prefarm

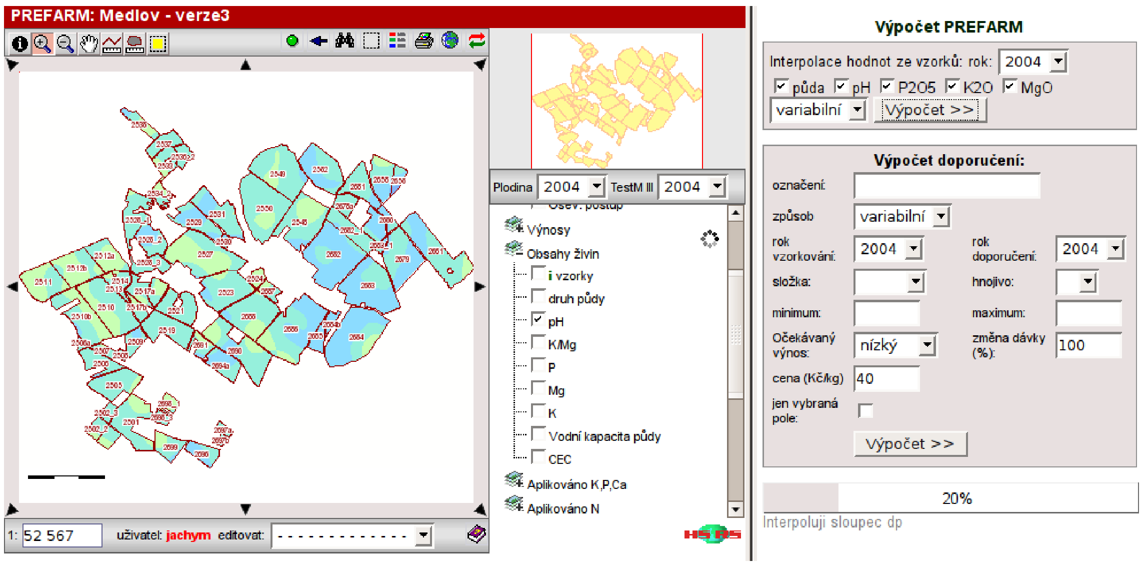
Prefarm is a project for providing optimal fertilization calculations. It is able to produce raster maps and tabular outputs, which can be used in the farmers decision process while finding the optimal fertilization variant. Output raster maps are later used in tractor computer together with GPS unit, so the fertilizer is distributed over the fields optimally.
Current version of Prefarm has two components:

1. Analytical part, which is done as plug-in for ESRI ArcView 4.0 program. The operator has to perform all necessary steps in order to get resulting maps and tabular data according to farmers needs. If some input parameter is changed by the farmer, the operator has to perform all calculations once again.
2. Visual part, which is done as web project using UMN MapServer for displaying maps, together with PHP and JavaScript. The farmer can check in the web browser calculation results and use the data in further decision process.

It is very complicated, or impossible to setup the analytical part of the system for more then one user. It is only hard imaginable to open the access to the application via Internet. Always there have to be person, which reacts at farmers needs and which do have to make the calculations manually.

In our currently developed version of Prefarm, we chose another approach, which is has also two main parts with one interlayer:

1. Analytical part is done as stand-alone scripts, using GRASS GIS tools. The scripts are wrapped in a file written in Python programming language. The scripts are doing special tasks, like creating raster maps for each nutrient and filling the PostgreSQL database with tabular data.
2. Visual part, which is done as web project using UMN MapServer for displaying maps, together with PHP and JavaScript. The farmer can check in the web browser calculation results and use the data in further decision process as well as recalculate the analysis with different input values (maximal or minimal amount of fertilizer, different fertilizer composition, ...).



1: 52 567 uživatel: jachym editovat: - - - - - - - - - - - - -

Plodina 2004 TestM III 2004

Interpolace hodnot ze vzorků: rok: 2004
[✓ půda] [✓ pH] [✓ P2O5] [✓ K2O] [✓ MgO]
variabilní

označení:			
způsob	variabilní		
rok vzorkování:	2004	rok doporučení:	2004
složka:		hnojivo:	
minimum:		maximum:	
Očekávaný výnos:	nízký	změna dávky (%):	100
cena (Kč/kg)	40		
jen vybraná pole:	☐		

20%

Fig. 5: *Prefarm application. Map view as well as control panel for nutrients interpolation and fertilization calculations.*

3. PyWPS as interlayer, which translates requests from the web application, runs server scripts, displays calculation progress and returns the calculation results back to the web browser.

This new solution, based on OGC standards, is multi-user system. Each user (farmer) can run her real-time analysis, modify input data directly, version control and management and other benefits. Operator is needed only for special cases, like user and fields management. As there is only one installation of the application, upgrades and bug fixes are easy to maintain.
Also from development point of view, as the application is modular (visualization part and calculation part, the calculation part has separated modules for each nutrient), we can work on it's parts without fear of breaking the whole system

4 Conclusion

Though WPS is relatively new standard, it is mature enough to be used in production applications. There are both, server and client implementations, which are actively developed. From usage of (OGC) standards do benefit both: users and code writers. Users (clients) can share data, informations and tools regardless who is providing the server solution. Code writers can make their applications easy interoperable and modularized.
In this paper, we introduced the standard and some of it's possible enhancements. Some of the standard implementations for both, client and server side, were described, with special attention to PyWPS.
Possible usage was demostrated on examples with PyWPS with both - proprietary (WPS Demo, Prefarm) and open clients (OpenLayers plugin).
Concrete benefits of WPS usage was demonstrated on Prefarm application, were current (old) solution together with new (developed) was compared.
Now days, applications are returning to the server-client concept. The WPS standard was the missing piece in the OGC Web Service standards, which makes it possible, to write thin client applications, which are using for analytical and data storage remote servers.

References

1. Open Geospatial Consortium Inc. *OpenGIS® Web Processing Service.* Version: 0.4.0. Document reference number: OGC 05-007r4. Editors: Peter Schut, Arliss Whiteside. 2005. http://www.opengeospatial.org/standards/requests/28
2. Michael C. and Ames, D. Evaluation of the OGC Web Processing
3. Service for Use in a Client-Side GIS
4. . *OSGeo Journal, volume 2.* 2007. Editor: Tyler Mitchell. ISSN 1994-1897. http://osgeo.org/journal
5. *Open Geospatial Consortium Inc. OpenGIS® Web Processing Service.* Version: 1.0.0. Document reference number: OGC 05-007r5. Editor: Peter Schut. 2007. http://www.opengeospatial.org/
6. Wikipedia contributors, Web Processing Service, *Wikipedia, The Free Encyclopedia,* http://en.wikipedia.org/w/index.php?title=Web_Processing_Service&oldid=168340145 (accessed November 13, 2007).
7. Wikipedia contributors, "SOAP," *Wikipedia, The Free Encyclopedia,* http://en.wikipedia.org/w/index.php?title=SOAP&oldid=169336421 (accessed November 13, 2007).
8. Wikipedia contributors, "Web Services Description Language," *Wikipedia, The Free Encyclopedia,* http://en.wikipedia.org/w/index.php?title=Web_Services_Description_Language&oldid=170295564 (accessed November 13, 2007).
9. Help Service – Remote Sensing s.r.o. *WPS Demo.* http://www.bnhelp.cz/mapserv/wpsdemo/ (accessed November 13, 2007).
10. PyWPS Development team. *Python Web Processing Service (PyWPS).* http://pywps.wald.intevation.org (accessed November 13, 2007).
11. GRASS Development Team. *GRASS GIS 6.3*. 2007. http://grass.itc.it (accessed November 13, 2007).
12. Open Geospatial Consortium Inc. *OpenGIS® Web Services Common Specification.* Version: 1.0.0. Document reference number: OGC 05-008 . Editor: Arliss Whiteside. 2005. http://www.opengeospatial.org/standards/common