GRASS jde na web: PyWPS

Jáchym Čepický[1]

Ústav geoinformačních technologií, Lesnická a dřevařská fakulta, Mendelova zemědělská a lesnická univerzita v Brně, Zemědělská 3,
613 00, Brno, Česká republika
jachym.cepicky@centrum.cz
http://les-ejk.cz

Abstrakt Open Geospatial Consorcium (OGC) definuje ve svém dokumentu 05-007r4 způsob, jakým mají být po síti nabízeny geoprostorové operace. Tento článek představuje jednu z implementací tohoto standardu. Cílem PyWPS (Python Web Processing Service) je nabídnout nástroje GIS pomocí webového rozhraní. PyWPS nabízí uživateli prostředí pro tvorbu vlastních částí WPS - tzv. procesů - přístupnit pokud možno jednoduše funkce GISu GRASS a dalších nástrojů na síti Internet. PyWPS je aplikace napsaná v programovacím jazyku Python a v současnosti podporuje všechny tři typy požadavků: GetCapabilities, DescribeProcess a Execute. Od počátku vývoje je myšleno na to, že PyWPS by měl sloužit jako prostředí pro moduly GISu GRASS. V základním nastavení jsou všechny geoprostorové operace prováděny v dočasně založené location, a ta je po proběhnutí všech výpočtů smazána. Lze však definovat i v location existující, ve které jsou již vstupní data uložena. V každém případě je k location vytvořen i dočasný mapset, ve kterém všechny operace probíhají a který je nakonec smazán.
Pro vytvoření vlastního procesu nabízí PyWPS velmi jednoduchou cestu: je pouze potřeba vytvořit skript v jazyce Python s definicí procesu, jeho vstupních a výstupních parametrů a funkcí execute(), která je PyWPS zavolána. Uživatel-programátor může využít veškerý potenciál programovacího jazyku Python. Omezení se na volání modulů GRASSu není nutné - v podstatě lze používat jakýkoliv program, který lze ovládat pomocí příkazové řádky nebo nabízející rozhraní pro jazyk Python.
Při žádosti Execute, PyWPS nejprve nastaví všechny vstupní parametry procesu, stáhne eventuálně požadované vstupní mapy ze vzdálených serverů, nastaví některé důležité proměnné prostředí (LOCATION_NAME, MAPSET, ...) a zavolá funkci execute() procesu. Každý proces může být spuštěn asynchronně - PyWPS je o aktuálním stavu a celkovém průběhu informován. Pokud proces selže a vrátí neprázdný řetězec, je tento použit ve výsledné zprávě o chybě.
V současné době není podporován celý (návrh) standard WPS, ale pouze cca. 95 % tohoto standardu. Všechny výstupní XML soubory byly validovány. PyWPS se snaží ctít zásadu KISS – „Keep it simple, stupid.“
V současnosti existuje příkladová instalace PyWPS, která je dostupná na adrese http://www.bnhelp.cz/mapserv/wpsdemo. Je založena na příkladovém datasetu České republiky a byla připravena ve spolupráci s firmou Help Service Remote Sensing. Implementovány byly 4 geoprostorové

operace: Analýza nejkratší cesty nad vektorovou mapou hlavních silnic (modul GRASSu v.net.path), neřízená klasifikace družicového snímku (i.maxlik), analýza viditelnosti (r.los) a analýza povrchového odtoku vody (r.flow). Další operace, jako jsou interpolace rastrových map z bodových dat, reklasifikace rastrových map a další jsou testovány v jiné aplikaci bez veřejného přístupu.
PyWPS je použitelný program, který v nejbližších dnech bude vydán ve verzi 1.0. Noví uživatelé a vývojáři jsou srdečně zváni, aby se podíleli na vývoji této aplikace. Vývoj PyWPS byl započat za podpory německé Nadace pro životní prostředí (Deutsche Bundestifftung Umwelt).

Klíčová slova: Geoinformatika, Webové služby, OGC, WPS, Python, GIS, GRASS, Svobodný software

Abstract. OGC's Web Processing Service draft paper 05-007r4 is defining the way, geoprocessing operations should be offered over networks using Web Services. This paper introduces one of it's implementation. The target of PyWPS (Python Web Processing Service) is to offer GIS tools over web interface. PyWPS offers the user environment for writing own parts of WPS - Processes - bringing functionality of GIS GRASS and other geoprocessing tools should be as easy as possible.
PyWPS has been written in Python programming language and currently it supports GetCapabilities, DescribeProcess and Execute types of request, though it is still in very early stage of development. From the beginning, PyWPS should serve as an environment for GIS GRASS modules. By default, all geoprocessing operations are working in temporary GRASS Location and after the resulting map has been calculated, this Location is removed afterwards. It is possible to use an already existing GRASS location and its included maps or creating a new location, which will be removed at the end of the process. In both cases temporary mapsets are created.
To create its own process, PyWPS has a very easy-to-use way to do so: create a Python script with a definition of the process, its inputs and outputs and an execute() function, which will be called by PyWPS. All advantages of Python programming can be used within the PyWPS framework. Not only GRASS modules can be called, basically all text-oriented tools can be used.
When the Execute request is called, PyWPS will set all input parameters for the process, download and store eventually requested input maps, set some environment variables (LOCATION_NAME, MAPSET, ...) and call the execute() function of the process. Processes can work asynchronously, they can inform PyWPS about the progress (percent of the process done and eventualy message). If the process fails, execute will return a non-null value (usually string), this will be used as a message in the fail report.
Currently not the whole WPS (draft) standard has been implemented, however, about 95% of this standard is usable. GetCapabilities responce has been declared as valid XML, other responces (DescribeProcess and

Execute) are at the moment not valid according to their XML scheme. At the moment PyWPS is keept as simple CGI application - following the KISS-rule (keep it simple,stupid).
Currently there is a sample installation of PyWPS, which is accessable from the URL http://www.bnhelp.cz/mapserv/wpsdemo/index.php. It is based on a sample dataset of Czech Republic and with help of the Help Service Remote Sensing company, the following processes have been implemented: Shortest path analaysis on vector-roads map (v.net.path), automated image classification (unsupervised classification - i.maxlik), visibility analysis (r.los) and analysis of flowlines (r.flow). Other processes, such as interpolation of raster maps from input point data, raster reclassification and others are tested in other application with no public access.
Though usable, PyWPS is in early beta stage and developers and new users are welcome to contribute on this application. PyWPS has been developed with support of DBU (Deutsche Bundestifftung Umwelt).

Keywords: Geoinformatics, Web services, OGC, WPS, Python, GIS, GRASS, Free Software

1 Úvod

Přivést funkce desktopového geografického informačního systému na síť bývá problém. Desktopové GISy nejsou primárně určeny k tomu, aby byly spouštěny neinteraktivně. K jejich ovládání je většinou potřeba grafické uživatelské rozhraní. Jedná se o velké systémy, často zatížené restriktivní licencí. V neposlední řadě je potřeba navrhnout komunikační protokol, kterým bude komunikovat serverová část GISu s klientskou.

1.1 Open Geospatial Consorcium

Open Geospatial Consorcium (OGC) je mezinárodní organizace sdružující jak fyzické tak právnické osoby z celého světa. Mezi hlavní cíle sdružení patří tvorba standardů výměny dat a služeb. Momentálně (21.8.2006) má sdružení 321 členů, mezi které patří na příklad University of Minnesota, US National Oceanic and Atmospheric Administration (NOAA), Coastal Services Center, ESRI, Autodesk, Inc., Masaryk University, MIT, a další. Za členství v organizaci se však platí a do ,,diskuze“ se tak mohou vložit pouze její členové [8].

1.2 Webové služby

Open Geospatial Consorcium představuje koncept tzv. OGC webových služeb (OGC Web Services – OWS) – sadu standardů pro komunikaci programů v prostředí počítačových sítí. Softwarové systémy by měly být navrhovány tak, aby tuto síťovou interakci podporovaly [17]. Služba (service) je tedy provedení úlohy, která byla vyvolána klientskou aplikací, a která je vykonána serverem.

Mezi nejznámější OWS standardy patří zejména [9]:

– Catalog Service (CAT), definující rozhraní pro vyhledávání v heterogenních katalogových serverech.
– Web Coverage Service (WCS), popisuje rozhraní pro získávání požadovaných rastrových dat ze vzdálených serverů
– Web Feature Service (WFS), popisuje rozhraní pro získávání požadovaných vektorových dat ze vzdálených serverů
– Web Mapping Service (WMS), definuje komunikaci mezi serverem a klientem, na jejichž základě jsou vytvářeny mapové náhledy z mnoha heterogenních zdrojů map.
– ...

Jedním z posledních schválených standardů je i OGC Web Processing Service.

2 OpenGIS Web Processing Service

Specifikace Open GIS Web Processing Service (WPS) je zatím (1.10.2006) ve fázi tzv. návrhu a nebyla ještě konzorciem OGC přijmuta jako standard. Jeho verze 0.4.0 je definována v dokumentu číslo OGC 05-007 [10]. Standard popisuje způsob, jakým jsou nabízeny geoprostorové operace a způsob komunikace mezi klientskou aplikací a serverem. Protokol je založen na jazyku XML.

Podobně jako ostatní standardy, rozlišuje WPS tři typy žádostí[1]: `GetCapabilities`, `DescribeProcess` a `Execute`.

2.1 GetCapabilities

Je-li serveru předán parameter `request` s hodnotou `GetCapabilities`, odešle zpět ke klientovi dokument XML, stávající se ze dvou částí: V první části je server představen, jsou zde zmíněny kontaktní informace na administrátora a další údaje (např. výše poplatků).

Ve druhé části pak server vrací seznam operací, které nabízí spolu s jejich krátkým popisem. Příklad formulace žádosti a výsledného dokumentu je v části 8.

2.2 DescribeProcess

Dalším krokem před spuštěním vlastního výpočtu je zjištění vstupních a výstupních parametrů ke konrétnímu výpočtu – procesu. Na žádost `DescribeProcess` server vrátí XML, které formát a možnou podobu těchto parametrů popisuje. `DescribeProcess` lze vyvolat buď vhodnout formulací URL metodou GET, podobně jako `GetCapabilities`, nebo vstupním XML souborem.

WPS rozlišuje čtyři typy vstupně-výstupních dat:

– `ComplexValue` je datový soubor, který je přímo součástí vstupního nebo výstupního XML. Většinou se jedná o vektorová data ve formátu GML.

[1] Mluvíme-li zde o „žádosti“, myslíme tím speciální parametr běžné žádosti klienta na server, který nese označní `request`, ne o žádosti jako celku.

– `ComplexValueReference` je pouze odkaz na datový soubor ležící (většinou) na vzdáleném serveru. Většinou je používán pro rastrová data. Výhodou je, že se lze odkazovat buď na samostatné soubory, nebo na další webové služby (WFS, WCS, WMS nebo i WPS), které data v požadovaném formátu vrací.
– `LiteralValue` je nějaký prostý text, bez bližší specifikace. Lze jej použít pro jednorázové zadání vstupních hodnot, stejně jako pro předání textového souboru.
– `BoundingBoxValue` specikuje požadovaný rozsah, většinou zájmového území. Je zadáván jako dvojice souřadnic protilehlých rohů výřezu.

Klient má možnost na základě tohoto popisu formulovat vstupní požadavek a spustit tak požadovanou operaci se všemi potřebnými atributy. Zároveň už také ví, jak bude vypadat odpověď ze strany serveru. Příklad výsledného dokumentu je v části 9.

2.3 Execute

Žádost `Execute` spouští požadovaný proces se všemi potřebnými parametry. Kromě vstupních dat lze také několika parametry ovlivnit podobu odpovědi serveru – zda-li pošle zpět výsledná data nebo XML dokument, ke kterému budou výsledky připojeny, a také to, má-li být proces proveden asynchronně. V takovém případě je ihned po obdržení požadavku vráceno zpět XML s odkazem na adresu, na které se lze o průběhu výpočtu ,,dočíst více“. To je důležité především pro případy, kdy výpočet trvá několik hodin či dní a spojení mězi klientem a serverem bývá přerušeno.

Příklad vstupního a výstupního XML je v části 10.

3 PyWPS 1.0

Python Web Processing Service (PyWPS) je implementace specifikace OGC Web Processing Service ve verzi 0.4.0. Vývoj tohoto programu byl započat během zahraniční stáže. Celý standard (zatím) není implementován, ale můžeme říci, že je v současnosti implementován z 95 % a že všechny důležité kompomenty implementovány jsou. Jedná se CGI aplikaci napsanou v programovacím jazyce Python [11].

Od začátku vývoje je myšleno na to, že jako ,,pracovní nástroj“ pro PyWPS bude využit geografický informační systém GRASS [4], který má pro tuto úlohu mnoho předpokladů:

– Jedná se o desktopový GIS, který lze ovládat z příkazové řádky (CLI)
– V současné době disponuje zhruba 300 moduly pro práci s rastrovými a vektorovými daty a umožňuje jejich analýzu v 3D prostoru. Další moduly jsou přítomny v GRASS Wiki [5] a poslední vývoj naznačuje, že se v GRASSu dočkáme i 4D GISu.
– Jedná se o systém vyvýjený pod licencí GNU/GPL, takže nic nebrání jeho nasazení na webovém serveru a zpřístupnění většímu množství uživatelů.

Vedle GRASSu lze samozřejmě využít i celou řadu dalších programů, jenž lze ovládat z příkazové řádky, zejména nástroje odvozené z knihovny GDAL [15] (`gdalwarp, gdal_translate, gdal_merge.py, ...`), PROJ.4 [16] (`cs2cs, nad2nad, ...`) a na příklad i statistický program R [12], jehož rozšíření disponují množstvím geoprostorových nástrojů a funkcí.

Na aplikaci PyWPS tak lze pohlížet ze dvou úhlů:

1. Jedná se o aplikaci, která přináší funkci (především) GISu GRASS na síť.
2. Uživatel nepotřebuje žádný desktopový GIS (GRASS, ArcGIS, Idrisi, ...) a naopak webový prohlížeč tyto funkce zpřístupňuje.

3.1 PyWPS – Execute

Postup při vyvolání některého procesu nabízeného PyWPS je následující:

1. V prvním kroku jsou zkontrolovány všechny vstupní parametry, pokud některý z nich chybí, snaží se PyWPS dosadit na toto místo hodnotu výchozí. V závislosti na typu vstupu je následně proveda některá z dalších operací:
   - `LiteralValue` – je zkontrolováno, jestli vstupní hodnota nabývá povoleného rozsahu a je-li požadovaného typu (celé číslo, textový řetězec, ...)
   - `ComplexValue` – vstupní soubor, který je součástí vstupního XML je uložen do zvláštního souboru
   - `ComplexValueReference` – PyWPS se snaží stáhnout data ze zadané adresy a uloží je do zvláštního souboru.
   - `BoundingBoxValue` – data není potřeba nijak zvlášť kontrolovat
2. Je vytvořen dočasný adresář, který zároveň slouží jako dočasná GRASS Location [3].
3. Zavolá funkci `execute()` procesu a spustí tím vlastní výpočet
4. Na konci výpočtu formuluje výsledné XML
5. Smaže všechny dočasně založené adresáře s jejich soubory (location, mapset v existující location)
6. Vrátí výsledné XML klientské aplikaci.

Pokud je proces spuštěn asynchronně, vrátí klientovi okamžitě XML se zprávou o tom, že process byl přijmut a na jaké adrese lze kontrolovat jeho průběh.

3.2 Vložení vlastních procesů

PyWPS je dodáván pouze s minimální sadou procesů, které mají sloužit hlavně jako dokumentace a příklady k tomu, aby uživatel byl schopný rychle naprogramovat aplikaci podle svých potřeb a připojit ji do své instalace PyWPS.

Jakýkoliv vlastní program může být napsaný v libovolném programovacím jazyce, je potřeba definovat minimálně jeho rozhraní v programovacím jazyce Python. Proces je pro PyWPS samostatný skript v jazyce Python s jednou třídou `Process` a minimálně dvěma funkcemi této třídy, a to `__init__`, ve které

je popsán proces svým identifikátorem, abstraktem a jsou zde také definovány vstupní a výstupní parametry a `execute`, ve které probíhá vlastní výpočet. V této funkci pak lze pomocí modulu `os` volat příkazy operačnícho systému, což sice není nejefektivnější způsob [13], ale ve srovnání s následným spuštěním geoprostorových operací, se o příliš velké plýtvání systémovými prostředky nejedná.

PyWPS exportuje několik proměnných, které lze vrámci skriptu použít pro snazší programování:

- `self.grassenv` obsahující některé proměnné prostředí GRASSu, jak jméno aktuální location a mapsetu
- `self.DataInputs` obsahující hash vstupních údajů, jako je jména vstupních rastrových a vektorových souborů (vstupy `ComplexValue` a `ComplexValue-Reference`) nebo přímo textové hodnoty (`LiteralValue`)
- `self.DataOutputs`, do kterézto proměnné lze uložit výsledné hodnoty (jména souborů, textové řetězce).

4 PyWPS Demo

Ve spolupráci s firmou Help Service Remote Sensing byla spuštěna demo aplikace, která má za úkol testovat koncept PyWPS a ukázat WPS v akci. Součástí příkladu je i webový klient, který není součástí distribuce PyWPS. PyWPS je pouze serverové řešení. PyWPS demo nabízí tyto čtyři geoprostorové operace:

- Výpočet nejkrační cesty pomocí GRASS modulu `v.net.path`. Vstupem jsou dva páry souřadnic – počáteční bod a cílový bod a výstupem je mapa ve formátu GML
- Neřízená klasifikace snímku z družice Landsat. Nejdříve jsou stažena data a uložena lokálně na serveru a následně je provedena neřízená klasifikace, při které je snímek rozčleněn do kategorií. Počet kategorií je definován uživatelem a je tak z pohledu uživatele jediným vstupem. Ke klasifikaci rastrových dat je použit modul `i.maxlik`
- Analýza viditelnosti, provedená nad lokálně uloženým digitálním modelem terénu, za použití modulu `r.los`
- Analýza povrchového odtoku provedená nad lokálně uloženým digitálním modelem terénu, za použití modulu `r.flow`

Při programování aplikací je potřeba dbát na to, že zejména moduly GRASSu často vypisují zprávy o svém průběhu na standardní výstup. To je dáno především stářím vývoje GRASSu a tím, že některé moduly používají místo GRASSových funkcí v jazyce C `G_message()` pro výpis stavu a `G_warning` pro varování, standardní funkci jazyka C `fprintf()`, což vede k tomu, že webový server (nejčastěji Apache Web Server) [2] chybně interpretuje výstup z těchto modulů jako chybně zformulovanou hlavičku `Content-type`. Je proto potřeba dbát na to, aby u většiny modulů byl standardní výstup přesměrován pomocí `1>&2` na standard error a tím se těmto problémům předejde. V aktuální vývojové verzi GRASSu 6.3 se na odstranění tohoto problému intenzivně pracuje.

5 Klient

Vývoj klientské aplikace nebyl součástí PyWPS. PyWPS je pouze serverové řešení. Theodor Förster, který implementoval standard WPS v jazyce Java, naprogramoval zásuvný modul do programu Jump. Doufáme, že nezůstane u této jedné implementace, ale že se přidají hlavně webové aplikace, pracující doposud s programem UMN MapServer. Právě služba WPS může z těchto „prohlížeček" map udělat plnohodnotné GISové aplikace, které mohou své výpočty provádět na několika vzdálených serverech najednou.

6 Závěr

PyWPS 1.0 je první verze CGI aplikace, která implementuje standard OGC Web Processing Service ve verzi 0.4.0. Tím, že je od počátku myšleno na to, že jako pracovní nástroj pro geoprostorové analýzy bude využit GIS GRASS, přináší PyWPS funkce tohoto GISu na web.

V dalším vývoji se soustředíme kromě nutného odstraňování chyb a nedostatků na lepším propojení Pythonu a GRASSu. Možnost přistupovat na rastrová a vektorová data uložená v interním formátu GRASSu je zajisté velmi atraktivní a bude jistě zajímavé sledovat vývoj rozhraní SWIG pro Python a další jazyky.

Další zajímavou možností bude propojení s některým s 3D vizualizačním nástrojem, na příklad s knihovnou VTK [6], protože 3D vizualizační nástroje GRASSu (NVIZ) nelze ovládat z příkazové řádky.

PyWPS neřeší rovnoměrné rozložení procesů mezi více procesorů nebo serverů soustředěných do clusteru. Projekt Moebius [7] by mohl být v tomto směru určitým řešením. Problém GRASSu je, že většina starých modulů nevyužívá pro svůj běh vlákna, takže nedokáží rozložit výpočet na více procesorů. Nové moduly [1,14] však tuto technologii již využívají a jejich běh tak může být výrazně optimalizován a zrychlen.

Adresa projektu PyWPS je `http://pywps.wald.intevation.org`. Infrastrukturu pro distribuovaný vývoj poskytuje portál GForge, spravovaný firmou Intevation GmbH.

7 Poděkování

Vývoj PyWPS byl finančně podpořen Německou nadací pro životní prostředí (DBU) a vývoj by se neobešel bez podpory firem Help Service Remote Sensing, GDF-Hannover a Intevation. Poděkování patří také IGA MZLU, která vývoj podpořila interním grantovým projektem č. 49/2006 - „Propojení mapového serveru ŠLP Masarykův les Křtiny se svobodným GIS GRASS umožňující online analýzu".

Reference

1. Agarwal, P.: Massive Terrain Data Processing: Scalable Algorithms, Free and open source software for Geoinformatics, Lausanne, 2006, http://www.foss4g2006.org/contributionDisplay.py?contribId=195&sessionId=40&confId=1, 1.10.2006
2. Apache HTTP Server Project, http://httpd.apache.org/, 1.10.2006
3. Neteler, H. Mitasova, 2004. Open Source GIS: A GRASS GIS Approach. Second Edition, Kluwer Academic Publishers, Boston, Dordrecht, ISBN 1-4020-8064-6
4. GRASS Development Team (2006). Geographic Resources Analysis Support System (GRASS), GNU General Public License. ITC-irst, Trento, Italy, http://grass.itc.it
5. GRASS Wiki, http://grass.gdf-hannover.de, 1.10.2006
6. Kitware Inc.: The Visualization ToolKit (VTK), http://public.kitware.com/VTK/, 1.10.2006
7. Čepický, Kryštov, Machalová, Procházka, Procházková: Moebius: Virtual mapserver for data abstraction, Proceedings z 21st European Conference For ESRI Users, Athens, 2006, *in print*.
8. Open Geospatial Consorcium, http://opengeospatial.org, 28.8.2006
9. Open Geospatial Consorcium: OpenGIS Specifications (Standards), http://www.opengeospatial.org/standards, 1.10.2006
10. Open Geospatial Consorcium: Web Processing Service, 05-007r4, http://www.opengeospatial.org/standards/requests/28, 1.10.2006
11. Python programming language, http://www.python.org, 1.10.2006
12. R Development Core Team, 2005: R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0, http://www.R-project.org
13. Stones, R., Matthew, N. 2000: Linux, začínáme programovat, Computer Press, Praha, ISBN 80-7226-307-2
14. Toma, L.: TerraCost: Scalable Computation of Least-Cost-Path Surfaces, Free and open source software for Geoinformatics, Lausanne, 2006, http://www.foss4g2006.org/contributionDisplay.py?contribId=106&sessionId=40&confId=1, 1.10.2006
15. Warmerdam, F. 2006: Geospatial Data Abstraction Library, http://gdal.org, 1.10.2006
16. Warmerdam, F. 2006: PROJ.4 - Cartographic Projections Library, http://proj.maptools.org, 1.10.2006
17. Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Webservice, 1.10.2006

8 GetCapabilities

Příklad žádosti:
`http://server/wps?service=WPS&version=0.4.0&request=GetCapabilities`

Příklad odpovědi:

```
<?xml version="1.0" ?>
<Capabilities version="0.4.0" ... >
        <ows:ServiceIdentification>
                <ows:Title>Beispiel WPS server</ows:Title>
                <ows:Abstract>WPS fuer DBU</ows:Abstract>
                <ows:ServiceType>WPS</ows:ServiceType>
                <ows:Fees>free</ows:Fees>
        </ows:ServiceIdentification>
        <ows:ServiceProvider>
                <ows:ProviderName>DBU</ows:ProviderName>
                <ows:ServiceContact>
                        <ows:IndividualName>Jachym Cepicky</ows:IndividualName>
                        <ows:PositionName>Student</ows:PositionName>
                        ...
                </ows:ServiceContact>
        </ows:ServiceProvider>
        <ProcessOfferings>
                <Process processVersion="0.1">
                        <ows:Identifier>addvalue</ows:Identifier>
                        <ows:Title>Add some value to raster map</ows:Title>
                </Process>
                <Process processVersion="0.1">
                        <ows:Identifier>classify</ows:Identifier>
                        <ows:Title>Image classification</ows:Title>
                        <ows:Abstract>
                                GRASS processed imagery
                                classification. Only unsupervised
                                is supported at the moment.
                        </ows:Abstract>
                </Process>
                <Process processVersion="0.1">
                        <ows:Identifier>shortestpath</ows:Identifier>
                        <ows:Title>Shortest path</ows:Title>
                </Process>
        </ProcessOfferings>
</Capabilities>
```

9 DescribeProcess

Příklad žádost:
`http://server/wps?service=WPS&version=0.4.0&request=DescribeProcess&identifier=shortestp`
Příklad odpovědi:

```
<?xml version="1.0" ?>
<ProcessDescriptions ...>
    <ProcessDescription ...>
        <ows:Identifier>shortestpath</ows:Identifier>
        <ows:Title>Shortest path</ows:Title>
        <ows:Abstract>Find the shortes path on the roads
        map on Czech republic road network</ows:Abstract>
        <DataInputs>
            <Input>
                <ows:Identifier>x1</ows:Identifier>
                <ows:Title>Start x coordinate</ows:Title>
                <LiteralData>
                        <ows:AnyValue/>
                </LiteralData>
            </Input>
            <Input>
                <ows:Identifier>y1</ows:Identifier>
                <ows:Title>Start y coordinate</ows:Title>
                <LiteralData>
                        <ows:AnyValue/>
                </LiteralData>
            </Input>
        ...
        </DataInputs>
        <ProcessOutputs>
            <ows:Output>
                <ows:Identifier>output</ows:Identifier>
                <ows:Title>Resulting output map</ows:Title>
                <ComplexOutput defaultFormat="text/xml">
                    <SupportedComplexData>
                        <Format>
                                text/xml
                        </Format>
                    </SupportedComplexData>
                </ComplexOutput>
            </ows:Output>
        </ProcessOutputs>
    </ProcessDescription>
</ProcessDescriptions>
```

10 Execute

Příklad žádost:

```
http://www.bnhelp.cz/cgi-bin/wps.py?service=WPS&\
version=0.4.0&\
request=Execute&identifier=shortestpath
DataInputs=x1,3310018.025,y1,5553487.845,x2,3736561.110,y2,5523815.282
```

Příklad žádosti ve formátu XML:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<Execute service='wps' version='0.4.0' store='true' status='false'
    xmlns="http://www.opengeospatial.net/wps"
    xmlns:ows="http://www.opengeospatial.net/ows">
<ows:Identifier>shortestpath</ows:Identifier>
<DataInputs>
    <Input>
        <ows:Identifier>x1</ows:Identifier>
        <LiteralValue>3310018.025</LiteralValue>
    </Input>
    <Input>
        <ows:Identifier>y1</ows:Identifier>
        <LiteralValue>5553487.845</LiteralValue>
    </Input>
    ...
</DataInputs>
</Execute>
```

Příklad odpovědi:

```
<?xml version="1.0" ?>
<ExecuteResponse ...>
    <ows:Identifier>shortestpath</ows:Identifier>
    <Status>
            <ProcessSucceeded/>
    </Status>
    <ProcessOutputs>
        <Output>
            <ows:Identifier>map</ows:Identifier>
            <ows:Title>Resulting output map</ows:Title>
            <ComplexValueReference
             format="text/xml"
             ows:reference="http://server/wpsoutputs/output2-2006-8-21-14-54-42.xml"
             />
        </Output>
</ProcessOutputs>
</ExecuteResponse>
```

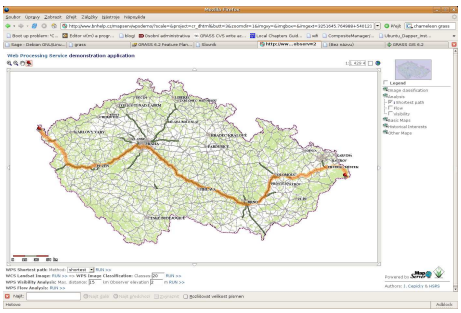
11 Příklad procesu

Příklad procesu tak, jak si jej může naprogrovat uživatel. Proces `addvalue` přidá zadanou hodnotu ke každé buňce vstupního rastrového souboru:

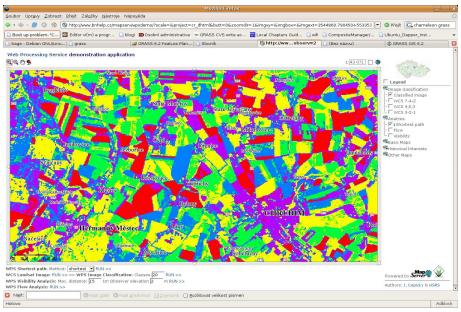
```
class Process:
    def __init__(self):

        self.Identifier = "addvalue"
        self.processVersion = "0.1"
        self.Title="Add some value to raster map"

        # Inputs
        self.Inputs = [
                    {
                        'Identifier':'input',
                        'Title': 'Input raster map',
                        'ComplexValueReference': {
                            'Formats': [ "image/tiff" ]
                        },
                    },
                    {
                        'Identifier': 'value',
                        'Title': 'Value to be added',
                        'LiteralValue': {'values':["*"]},
                        'dataType' : type(0.0),
                        'value':None
                    },
                ]

        # Output
        self.Outputs = [
                {
                    'Identifier': 'outputRef',
                    'Title': 'Resulting output map',
                    'ComplexValueReference': {
                        'Formats':["image/tiff"],
                        },
                    'value':None
                },
            ]

        # storeSuport should the resulting map be stored on our disk?
        self.storeSupported = "true"

        self.statusSupported = "true"
```

```
def execute(self):
    # 1 - import dat
    os.system("r.in.gdal -o in=%s out=input >&2" %\
            (self.DataInputs['input']))
    self.status = ["data naimportovana", 10]

    # 2 - nastaveni pracovniho regionu
    os.system("""g.region rast=input >&2""")

    # 3 - pridani hodnoty
    os.system("r.mapcalc output=input+%f >&2" % \
            (float(self.DataInputs['value'])))

    # 4 - export dat do formatu geotiff
    os.system("r.out.gdal type=Int32 in=output out=%s 1>&2" % "output.tif")
    self.status = ["data vyexportovana", 90]

    # 5 - nastaveni vystupnich parametru
    if "output.tif" in os.listdir(os.curdir):
        self.DataOutputs['outputRef'] = "output.tif"
        return
    else:
        return "Output file not created!"
```

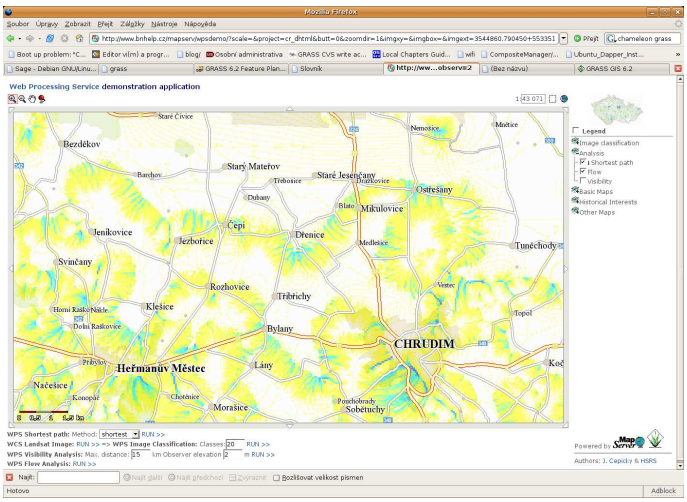
12 Obrazové přílohy



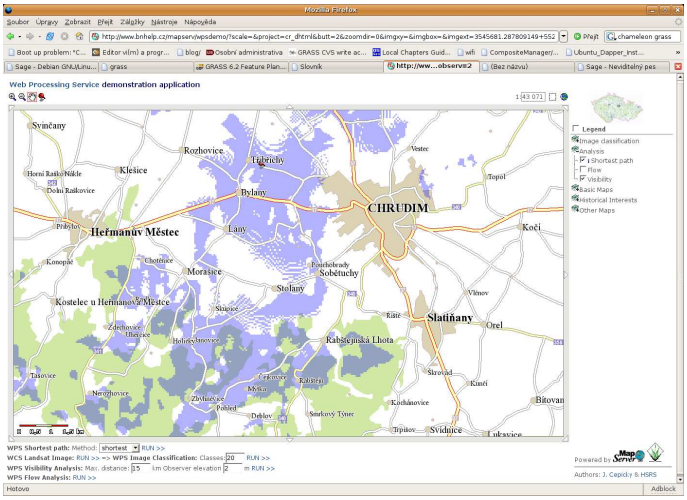
Obrázek 1. Příklad výpočtu nejkratší cesty realizovaného pomocí PyWPS. Pro výpočet byl použit modul GISu GRASS `v.net.path`. Výsledek je zobrazen ve standradním mapovém okně webové mapové aplikace. Zdroj: `http://www.bnhelp.cz/mapserv/wpsdemo/`



Obrázek 2. Příklad automatické klasifikace. Pro výpočet byl použit modul GISu GRASS `i.maxlik` vstupní snímek družice LANDSAT TM (stažený ze vzdáleného serveru) byl klasifikován na pět tříd. Výsledek je zobrazen ve standradním mapovém okně webové mapové aplikace. Zdroj: `http://www.bnhelp.cz/mapserv/wpsdemo/`



Obrázek 3. Výpočet hustoty povrchového odtoku z digitálního modelu terénu uloženého na serveru modulem `r.flow`. Výsledek je zobrazen ve standradním mapovém okně webové mapové aplikace. Zdroj: `http://www.bnhelp.cz/mapserv/wpsdemo/`



Obrázek 4. Analýza viditelnosti nad digitálním modelem terénu uloženým na serveru modulem `r.los` (z okolí obce Třibřichy). Výsledek je zobrazen ve standradním mapovém okně webové mapové aplikace. Zdroj: `http://www.bnhelp.cz/mapserv/wpsdemo/`