Embrio and Wuiw

two web interfaces for WPS

Lorenzo Becchi (ominiverdi.org)

1:00 pm, September 26, 2007

Adapted from Jachym Cepický (GRASS goes web: PyWPS 0.1)
http://Les-ejk.cz
Embrio

- Technologies: DHTML + AJAX + PHP/Mapscript
- Scope: an interface based on UMN Mapserver Mapscript to access GRASS functions (Mapserver and WPS on the same server).
- Dev state: beta

Wuiw

- Tecnologie: DHTML + AJAX
- Scope: an interface as man in the middle between a WPS server and other OWS servers (WCS, WFS) joining remote resources for data and processing.
- Dev state: alpha
PyWPS in action - WPS Demo

- http://pywps.ominiverdi.org/
  - Repository of demo apps created by ominiverdi.org
  - Developers:
    - Lorenzo Becchi (ominoverde)
    - Luca Casagrande (doktoreas)
  - All applications shown have to be considered under development. If you find a bug, please, post it to PyWPS's mailing list: pywps-devel@wald.intevation.org
EMBRIIO - a simple PyWPS AJAX Web Interface

This page is intended to show the development status of a Web User Interface for PyWPS

Demo applications

- **V.Buffer** - Create a buffer around features of given type (areas must contain centroid).
- **R.Los** - Generates a raster map output in which the cells that are visible from a user-specified observer location are marked with integer values that represent the vertical angle (in degrees) required to see those cells (viewshed).
- **V.Net.Path** - Find shortest path on vector network.
- **R.walk and r.drain** Find shortest path between 2 points using slope factor as cost value.

Demo applications of ka-Map tool

This applications use ka-Map API to create tiled cache and map navigation. SLD style definition is supported for the WPS output layer.

- **R.Los with Embrio interface**
- **R.Los with Winman interface**

Developers

- doktorea
- ominiverdi
GRASS Buffer module (v.buffer)

Create a buffer around features of given type (areas must contain centroid).

Usage: click on map to set coordinates of the centroid, manually insert the radius param.
Then click on Go! button to see overlayed output.

**Params**

<table>
<thead>
<tr>
<th>X:</th>
<th>602233</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y:</td>
<td>4921471</td>
</tr>
<tr>
<td>Radius:</td>
<td>3000</td>
</tr>
</tbody>
</table>

[Go!]
GRASS Buffer module

r.los generates a raster map output in which the cells that are visible from a user-specified observer location are marked with integer values that represent the vertical angle (in degrees) required to see those cells (viewshed).

Usage: click on map to set coords. Use selects to change distance and theta params. Then click on Go! button to see overlaid output.

<table>
<thead>
<tr>
<th>Params</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X:</td>
<td>592614</td>
</tr>
<tr>
<td>Y:</td>
<td>4926515</td>
</tr>
<tr>
<td>Max distance where check visibility:</td>
<td>(range 500 -&gt; 1500)</td>
</tr>
<tr>
<td>1500</td>
<td></td>
</tr>
<tr>
<td>Height of the observer:</td>
<td>(range 1 -&gt; 3)</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Embrio - V.Net.Path

GRASS Routing module (v.net.path)
Find shortest path on vector network.
Usage: click on map to set coords of start and end point. Then click on Go! button to see overlayed output.

<table>
<thead>
<tr>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1: 591061</td>
</tr>
<tr>
<td>Y1: 4926622</td>
</tr>
<tr>
<td>X2: 605142</td>
</tr>
<tr>
<td>Y2: 4920243</td>
</tr>
<tr>
<td>Cost: 0</td>
</tr>
</tbody>
</table>

http://pywps.ominiverdi.org/subversion/trunk/web/embrio/vector/v_net_path/v_net_p
GRASS Minor cost path module (using r.walk)

Find shortest path between 2 points using slope factor as cost value.

Usage: click on map to set coordinates of start and end point. Then click on Go! button to see overlaid output.

Params:
- X1: 590962
- Y1: 4926435
- X2: 600051
- Y2: 4918429

Go!
Embrio + ka-Map - R.Los
Embrio + ka-Map + Winman - R.Los
Embrio – last prototype
• PyWPS in action - WUIW Demo

  - Code status: Beta
  - Developers:
    - Jachym Cepicky (jachym)
  - Based on OpenLayers API, GRASS, Mapserver
• PyWPS in action - WUIW Demo
Open Geospatial Consortium, Inc (OGC)

- International voluntary consensus standards organization
- Development and implementation of standards for geospatial content and services
- Active Members: 350 (Sep 2007) – University of Minnesota, US National Oceanic and Atmospheric Administration (NOAA) Coastal Service Center, ESRI, Autodesk Inc., MIT, ...
- http://www.opengeospatial.org/
Web service

- Software system designed to support interoperable machine-to-machine interaction over a network (Wikipedia)
- In OGC terminology, "Service" refers to a processing task that is invoked by a client and executed by a server, usually across a network.
- The OpenGIS Specifications that make this possible are referred to as "OGC Web Services".
- OpenGIS Web Service (OWS):
  - OpenGIS Catalog Service (CAT)
  - OpenGIS Web Coverage Service (WCS)
  - OpenGIS Web Feature Service (WFS)
  - OpenGIS Web Map Service (WMS)
  - . . .
  - Web Processing Service (WPS) (draft)
OpenGIS Web Processing Service

- Document OGC 05-007r4, version 0.4.0
- Not yet OGC standard, "Discussion Paper", Draft
- To offer any sort of GIS functionality to clients across a network
- XML-based communication protocol

Es: http://pywps.ominiverdi.org/cgi-bin/wps?service=WPS&version=0.4.0&request=...
• WPS request=GetCapabilities

http://www.bnhelp.cz/cgi-bin/wps?service=WPS&version=0.4.0&request=GetCapabilities

<?xml version="1.0" ?>
<Capabilities version="0.4.0" ... >
  <ows:ServiceIdentification>
    <ows:Title>Sample WPS server</ows:Title>
    <ows:Abstract>WPS for Lausanne</ows:Abstract>
    <ows:ServiceType>WPS</ows:ServiceType>
    <ows:Fees>free</ows:Fees>
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
    <ows:ProviderName>GDF</ows:ProviderName>
    <ows:ServiceContact>
      <ows:IndividualName>Jachym Cepicky</ows:IndividualName>
      <ows:PositionName>Student</ows:PositionName>
    </ows:ServiceContact>
    ... 
  </ows:ServiceProvider>
  ...
</Capabilities>
<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>
- WPS request=DescribeProcess

http://www.bnhelp.cz/cgi-bin/wps.py?service=WPS&version=0.4.0&
request=DescribeProcess&identifier=addvalue

```xml
<?xml version="1.0" ?>
  <ProcessDescriptions ...>
    <ProcessDescription ...>
      <ows:Identifier>addvalue</ows:Identifier>
      <ows:Title>Add value</ows:Title>
      <ows:Abstract>Adds some value to each cell of input raster map</ows:Abstract>
      <DataInputs>
        <Input>
          <ows:Identifier>value</ows:Identifier>
          <ows:Title>Value to be added</ows:Title>
          <LiteralData>
            <AllowedValues>
              <Value>1</Value>
              ...
            </AllowedValues>
            <ows:DefaultValue>10</ows:DefaultValue>
          </LiteralData>
        </Input>
        ...
      </DataInputs>
    </ProcessDescription>
  </ProcessDescriptions>
```

**WPS request=DescribeProcess**

```xml
<Input>
  <ows:Identifier>map</ows:Identifier>
  <ows:Title>Input raster map</ows:Title>
  <ComplexData defaultFormat="image/tiff">
  </ComplexData>
</Input>
</DataInputs>

<ProcessOutputs>
  <Output>
    <ows:Identifier>value</ows:Identifier>
    <ows:Title>literal value + 1</ows:Title>
    <LiteralOutput> ... </LiteralOutput>
  </Output>
  <Output>
    <ows:Identifier>map</ows:Identifier>
    <ows:Title>Resulting output map</ows:Title>
    <ComplexOutput defaultFormat="image/tiff">
      ...
    </ComplexOutput>
  </Output>
</ProcessOutputs>
</ProcessDescription>
</ProcessDescriptions>
```
http://www.bnhelp.cz/cgi-bin/wps.py?service=WPS&version=0.4.0&request=Execute&identifier=addvalue&DataInputs=value,5,map,http://localhost/data/soils.tif

<?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>addvalue</ows:Identifier>
<DataInputs>
<Input>
<ows:Identifier>value</ows:Identifier>
<LiteralValue>5</LiteralValue>
</Input>
<Input>
<ows:Identifier>map</ows:Identifier>
<ComplexValueReference reference="http://localhost/data/soils.tif" />
</Input>
...
</DataInputs>
</Execute>
<?xml version="1.0" ?>

<ExecuteResponse ...>
  <ows:Identifier>addvalue</ows:Identifier>
  <Status>
    <ProcessSucceeded/>
  </Status>
  <ProcessOutputs>
    <Output>
      <ows:Identifier>value</ows:Identifier>
      <ows:Title>literal value + 1</ows:Title>
      <LiteralValue>6</LiteralValue>
    </Output>
    <Output>
      <ows:Identifier>value</ows:Identifier>
      <ows:Title>Resulting output map</ows:Title>
      <ComplexValueReference format="image/tiff"
    </Output>
  </ProcessOutputs>
</ExecuteResponse>
PyWPS 2.0

- Implementation OGC's WPS standard (90-95 %)
- CGI Application
- Python programming language
PyWPS starts as dedicated connector to GRASS

- CLI
- More than 300 modules for raster and vector analysis
- GNU/GPL
- GRASS Functionality can be via PyWPS offered in Internet

User does not need Desktop-GIS (GRASS, ESRI, Idrisi, ... ) – Web browser becomes GIS

One can use other CLI-oriented programs (PROJ.4, GDAL, R, ... )
PyWPS 2.0 – Execute – how it works

- Controlling input data, if all necessary parameters have arrived (Identifier, DataInputs, . . .)
- Loading process, for each input:
  - LiteralValue: Controlling, if input fits AllowedValues array
  - ComplexValue: Embed input files will be extruded from input XML request into separate files
  - ComplexValueReference: Tries to download the data from external source and stores it to new file
  - BoundingBoxValue
- If some DataInput is missing, it looks for the default value value
PyWPS 2.0 – Execute – how it works

- Creates temporary GRASS Location or just temporary Mapset within existing location, which will be deleted, after the work is done.
- Calls function execute() of the process.
- Formulates output XML file.
- Deletes temporary files (location, mapset, pid file).
- Returns output XML or resulting map file (TIFF, GML) to the client.
- Process can be run asynchronously: After the request is accepted, XML response is immediately returned with <ProcessAccepted /> element and the calculation is forked to background.
• PyWPS 2.0 - Addvalue – Sample process

**Inputs**
- Literal input: il valore da aggiungere
- ComplexValueReference input: una mappa raster

**Outputs**
- Literal output: Input+1
- ComplexValueReference: la mappa risultante (GeoTIFF)
2006-06-01

PyWPS 2.0 – Sample process

```python
01 class Process:
02     def __init__(self):
03         self.Identifier = "addvalue"
04         self.Title = "Sample process for demonstration purposes"
05         self.Inputs = [
06             { # 0
07                 'Identifier': 'value',
08                 'Title': 'Value to added',
09                 'LiteralValue': {'values': [0, 1, 2, 3, 4, 5]},
10                 'dataType': type(0),
11                 'value': 0, # default},
12             { # 1
13                 'Identifier': 'map',
14                 'Title': 'The raster map',
15                 'ComplexValueReference': {'Formats': ['image/tiff']}
16             }... ]
17         self.Outputs = [
18             { # 0
19                 'Identifier': 'value',
20                 'Title': 'Input value + 1',
21                 'LiteralValue': {},
22                 'value': 1 }... ]
```
def execute(self):
    self.status = ["The start", 5]
    self.Outputs[0]["value"] = self.Inputs[0]["value"]+1
    self.status = ["LiteralValue set", 20]

    self.status = ["Data import", 25]
    os.system("r.in.gdal in=%s out=map" % (self.Inputs[1]["value"]

    self.status = ["Creating output map", 50]
    os.system("r.mapcalc map=map+%d" % (self.Inputs[0]["value"]))

    self.status = ["Exporting map", 75]
    if os.system("r.out.gdal in=map out=output.tif type=UInt16 >
    return "Could not export map"
else: # ok
    return
Conclusions

- WPS Standard implemented to usable degree
- Making GRASS scripts run via web-interface was never easier
- It is relatively simple to connect UMN MapServer (or ARC IMS) with GRASS via PyWPS. Further GRASS development will make this even easier
● PyWPS

● Process definition (data inputs and outputs) is primitive – build set of classes for process definition

● GRASS

   ● Implementation of new GRASS-python interface (Alessandro Frigeri aka ’geoalf’)

   ● 3D views via VTK (Sören Gebbert aka ’huhabla’)

● Embrio & Wuiw

   ● Road to stable versions
## Attribution-NonCommercial-ShareAlike 2.5

**You are free:**
- to Share — to copy, distribute, display, and perform the work
- to Remix — to make derivative works

**Under the following conditions:**

- **Attribution.** You must attribute the work in the manner specified by the author or licensor.
- **Noncommercial.** You may not use this work for commercial purposes.
- **Share Alike.** If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

---

Your fair use and other rights are in no way affected by the above. This is a human-readable summary of the Legal Code (the full license).