OSGeo Journal

The Journal of the Open Source Geospatial Foundation

Volume 1 / May 2007

In This Volume

OSGeo

Your Open Source Compass

Editorial

Editorial

Our New Publication

Welcome to Volume 1 of our brand new journal! This new publication celebrates the many successes in the general open source and geospatial communities as well as those particular to the Open Source Geospatial Foundation (OSGeo)[1].

The goal of the journal is to update, inform and educate all kinds of readers. It will cover project related news updates, case studies, tutorials and more. In particular, it will help capture OSGeo project updates and their plans for the future.

This volume was planned to replace and expand the retired GRASS-News[2] publication – which provided strong technical tutorials and community interaction. This new journal will continue to include GRASS related information along with many other projects - regardless of whether they are inside or outside of OSGeo's umbrella.

Call For Articles

You can help by providing articles or ideas for publication in the next volume. There is also room for more editors, layout designers and reviewers. To get involved complete the Journal feedback form[3] on the OSGeo website.

The next volume will continue to have a broad approach to the types of articles it publishes, including programming tutorials, introducing new projects, describing case studies, project updates, and more. Articles are grouped into areas by content type (e.g. tutorials, news, studies, etc.) which will help expose readers to a variety of domains and tools.

It has been a pleasure to work with the rest of the editorial team and contributing writers. Their hard work in finding content, writing articles, collating material, proofing and formatting has helped bring forth some very interesting material in a very short period of time. Needless to say, I'm looking forward to Volume 2 later in the year. Hopefully just in time for the **OSGeo FOSS4G annual conference**[4].

Tyler Mitchell
Editor in Chief
http://osgeo.org
tmitchell AT osgeo.org

[1]Open Source Geospatial Foundation (OSGeo) web site: http://www.osgeo.org
[2]GRASS-News web site: http://grass.itc.it/newsletter/index.php
[3]Journal feedback form: http://www.osgeo.org/feedback/Journal
[4]FOSS4G 2007 web site: http://www.foss4g2007.org

Contents of this volume:

2007 FREE AND OPEN SOURCE SOFTWARE
FOR GEOSPATIAL (FOSS4G) CONFERENCE
VICTORIA CANADA SEPTEMBER 24 TO 27, 2007

FOSS4G - Conference Registration Open

We are pleased to announce that online registration for the 2007 Free and Open Source Software for Geospatial conference (FOSS4G 2007) is now open. FOSS4G 2007 is the annual event bringing together the people and companies who create, use, and support open source geospatial software. Register now online.[5]

Register prior to the July 27th Early Bird Deadline to save on your registration fees! Take advantage of the opportunity FOSS4G 2007 offers to network with fellow geospatial data professionals, renew old acquaintances and make new ones.

For up-to-date information, registration and/or to submit a presentation, please visit the conference website.[6]

EXHIBITOR & SPONSORSHIP OPPORTUNITIES

For information on exhibitor and sponsorship opportunities, see the sponsors page[7] or contact Paul Ramsey, Conference Chair via email.[8]

SUBMIT A PRESENTATION

You can submit a presentation online.[9] The deadline for submissions is June 29th 2007.

FOSS4G presentations are 25 minute talks, with 5 minute question and answer sessions at the end. Presentations cover the use or development of open source geospatial software. Anyone can submit a presentation proposal and take part in the conference as a presenter. More information is available on the presentations page on the website.

We hope to see you in Victoria, Canada in September!

[5]Register online at: http://www.foss4g2007.org/register/
[6]Conference website: http://www.foss4g2007.org/
[7]Sponsors page: http://foss4g2007.org/sponsors
[8]Email Paul Ramsey at: pramsey@foss4g2007.org
[9]Submit a presentation at http://www.foss4g2007.org/presentations/

News

Brief Notes

A collection of recent News and Announcements

Compiled by Jason Fournier

OSGeo's 1st Anniversary

February 4th, 2007 was the first anniversary of OSGeo. At that time 25 people met face-to-face, and many more via phone and IRC, to discuss the possibility of starting an umbrella organisation. These participants represented over 13 different open source projects. The foundational purpose of the organisation was to help promote and continue to develop open source tools in the geospatial sphere. Since that time, much has happened and momentum around OSGeo continues to develop.

For an overview of many of the highlights from the first year please visit Tyler's blog.[10]

OSGeo Accepted for Google Summer of Code 2007

OSGeo is pleased to announce that Google has accepted OSGeo as a mentoring organization for the Google Summer of Code 2007 program. This program provides funding for students to work on open source projects under the support of experienced mentors. The projects participating through OSGeo are GDAL, GeoServer, GeoTools, GRASS, MapGuide, PostGIS, uDig, OpenJUMP and MapServer. Students interested in participating can find more information on OSGeo project ideas on the OSGeo wiki[11] and more information about the program as a whole at on Google's SOC site [12]

Source: OSGeo-Announce list and OSGeo website[13]

New Sponsors

In addition to continuing Sustaining Sponsor Autodesk, new Associate Sponsors include Lizardtech, 1Spatial and First Base Solutions:

- Autodesk - http://www.autodesk.com
- Lizardtech - http://lizardtech.com
- 1Spatial - http://www.1spatial.com
- First Base Solutions - http://www.firstbasesolutions.com

[10]Tyler's anniversary blog article: http://www.osgeo.org/tyler/osgeo_1st_anniversary
[11]Summer of code project ideas: http://wiki.osgeo.org/index.php/2007_SoC_Merged_Ideas
[12]SOC web site: http://code.google.com/soc/
[13]SOC online announcement: http://osgeo.org/news/soc2007

GDAL Sponsors

There are also several sponsors who have provided support for GDAL/OGR development through OSGeo's sponsorship programme.

PROJECT NEWS

This section includes brief updates about various projects. For more specific details see the Developer Announcements section on page 57.

MapGuide

March 2007 - MapGuide graduated from OSGeo Incubation, showing an acceptable level of participation, freedom from code encumbrances, and use of standard open source development methodologies. MapGuide Open Source is currently at version 1.1.0.

MapServer

The MapServer development team plans to release version 5.0 by the end of summer 2007. The 5.0 Release Plan outlines a number of planned updates, including:

- Support for AGG, an up and coming image renderer as an alternative to GD
- Upgrade of WMS support to version 1.3.0
- Improved OGC SOS Server support
- New OGC OWS Common support version 1.0.0
- Dynamic charting capability
- Label prioritization
- MapScript memory management fixes
- Redesign of the LOG/DEBUG output mechanism

The MapServer Release Plan[14] for version 5.0 is online.

Mapbender

Mapbender has recently announced a series of new features as well as future roadmap for new developments, including:

- Integration of OpenLayers as an optional Mapbender module is scheduled for 2007
- WFS-T / Digitizing enhancement: handling of complex geometries
- Wider use of AJAX /JSON and OO technology: improvement of module interfaces to do justice to wider development audience
- Evaluation of packing algorithms: due to traffic minimization, there are few source code comments. JSDoc and PHPDoc will be used to comment in the future. The traffic issue will be resolved by extracting the comments prior to deployment and by packing JS-code.

GDAL/OGR

GDAL/OGR has announced it's 1.4.1 release this past quarter. It has launched a Sponsorship program through OSGeo. The first sponsors are Analytical Graphics, Applied Coherent Technology, SRC, Safe Software, Cadcorp, Waypoint, MicroImages and i-Cubed.

GeoNetwork

Update on developments in the first Quarter of 2007. Work has been concentrated on a couple of aspects of the project lately. They are:

- Development of GeoNetwork opensource version 2.1 resulting in beta releases
- Upgrading of the Community website
- Work related to the OSGEO incubation process

GRASS GIS

Project Update 2007 - Q1:

- Italian GRASS and GFOSS Users Meeting - GRASS and GFOSS Users Meeting, Palermo (Italy), 14-16 Feb 2007
- 12 Feb 2007: New GRASS bug and wish tracker - Gforge based
- 10 Feb 2007: GRASS GIS 6.2.1 winGRASS/Cygwin binaries available

Quantum GIS

The QGIS project officially enter OSGeo incubation in March 2007. This starts the incubation process to bring QGIS into full OSGeo membership. OSGeo is currently incubating a number of projects and QGIS joins GRASS and OSSIM in the Desktop Applications category.

[14]MapServer Release Plan: `http://mapserver.gis.umn.edu/development/release_plans/mapserver_5_0`

UPCOMING EVENTS

These are only a few of the upcoming events that OSGeo and its members will be presence at over the next few months.

III MapServer Users National Meeting, Brasilia

May 9-11, 2007 This is the MapServer and open source geospatial conference held in Brasilia, Brazil. See `http://www7.univali.br/elis2/enum07/` for English and for Portuguese information.

Where 2.0, San Jose

May 29-30, 2007 The O'Reilly Where 2.0 conference held in San Jose, California, USA. More information: `http://conferences.oreillynet.com/where2007/`.

OSCON 2007, Portland

July 23-27, 2007 The O'Reilly Open Source Convention held in Portland, Oregon, USA. More information: `http://conferences.oreillynet.com/os2007/`.

Map Asia 2007, Kuala Lumpur

August 14-16, 2007 The 6th Annual International Conference and Exhibition on Geographical Information Technology and Applications. The conference theme is *Maponomics — Economic Growth Thorough Geo-Information Technology*. Map Asia leverages an international initiative aimed to provide an apt platform for the convergence, sharing and use of Geospatial technologies. Founded in the year 2002, Map Asia is the largest Annual Asian Conference and Exhibition in the field of GIS, GPS, Aerial Photography and Remote Sensing. Held in Kuala Lumpur, Malayasia. More information: `http://www.mapasia.org`.

FOSS4G 2007, Victoria

September 24-27, 2007 The annual Free and Open Source Software for Geospatial (FOSS4G) conference brings together the people who create, use, and support open spatial software. No other event brings together members of the open source development, open data creation, and open standards promotion communities like FOSS4G. The event will be held in Victoria, British Columbia, Canada. For more information please visit the FOSS4G 2007 homepage: `http://www.foss4g2007.org/`.

INTERGEO, Leipzig

September 25-27, 2007 Conference and trade fair for geodesy, geoinformation and land management held in Leipzig, Germany. They expect over 17,000 visitors from more than 80 countries. More information at: `http://www.intergeo.de`.

BOARD

Highlights from recent OSGeo Board of Directors meetings:

- Confirmed Howard Butler as new SAC chair
- Approved formation of New Mexico USA local chapter
- Approved motion for MapGuide OS to graduate from Incubation
- Approved motion for QGIS and FDO to enter Incubation
- Support OSGeo participation in the Google Summer of Code
- Appointed Tyler as OSGeo secretary to replace Rich, and as Chief Returning Officer for to coordinate/monitor elections
- Approved Board and Membership Election Procedures
- Tasked CRO to start with Charter Membership election process
- Financial review

Event Reports

North Carolina OSGeo User Group Meets

Following the 2007 NC GIS Conference

by Julia Harrell

Bi-annually, North Carolina GIS professionals hold a statewide GIS conference to showcase their many exemplary projects and initiatives. This year was the 20th anniversary of the NC GIS Conference, one of the nations oldest state-level GIS events, and had over 950 attendees, 159 presenters, and 58 exhibitors

On March 1st, a small group of dedicated, open-minded GIS professionals led by Dr. Helena Mitasova (of the renowned GRASS project fame) held the first OSGeo affiliated NC Open Source GIS user group meeting at the 2007 North Carolina GIS Conference in Winston Salem, NC, USA.

There were approximately 20 attendees, with a majority of the group who signed up for a future listserve being of the University persuasion. There were also a handful of Federal, State and Local Government representatives. Two staff members from an enormous commercial GIS software provider also attended.

Figure 1: Helena introduces OSGeo and gives a tour of the OSGeo wiki site

The informal evening meeting was held after the main GIS conference, and the agenda consisted of the following topics:

Welcome (*Julia Harrell, GIS Coordinator, NC Department of Environment and Natural Resources - NC DENR*)

Short introduction to the OSGeo organization and discussion of the NC Affiliated OSGeo user group and its mission (*Dr. Helena Mitasova, Research Associate Professor with North Carolina State University Department of Marine Earth and Atmospheric Sciences*)

Short presentation and demo of QGIS and uDig Open Source Desktop GIS Software (*Doug Newcomb,*

System Administrator with the US Fish and Wildlife Services Raleigh, NC Office)
Discussion of potential Open Source GIS Curriculum offerings and a 1-day, hands-on Open Source GIS workshop later this summer (*Rodney Jackson, Director of the Geospatial Technology Corporate and Continuing Education Program at Central Piedmont Community College in Charlotte, NC*)
Briefing on upcoming 2nd Annual Open Source GIS Conference (OSG-SD) at San Diego State University, and a recap of last years event (*Dana Nibby, GIS Database Analyst with the NC Department of Transportation GIS Unit*)
Demo of GAIA3 for Incident Response, a free and interoperable application created with a 2006 FGDC CAP grant by The Carbon Project, NC DENR, the City of Charlotte, and Wake County GIS Departments (*Jeff Harrison, President & CEO, The Carbon Project*)
Announcement of the next day's Open Source enhanced GIS Conference presentations (*Julia Harrell, GIS Coordinator, NC DENR*)
Texas Mesonet: Processing Large Datasets in Real Time (*Gerry Creager, Texas A&M University*)
Geospatial Web Services and Scientific Workflows for Distributing and Modeling Marine Mammal Data (*Dr. Patrick Halpin, Duke University*)
Developing the NCCOOS Coastal Data Viewer: Challeges in Real Time Marine GIS (*Jesse Cleary, UNC Chapel Hill*)
OpenGIS Web Services and Service Oriented Architecture (*Tobin Bradley, GISP, Mecklenburg County, NC*)

Figure 2: NC OSGeo User Group Attendees

Low Cost High Value Web Mapping

A Whirlwind Survey of Free and Open Source GIS Solutions

Panel of presenters:

PostgresSQL/PostGIS: Spatially Enabled Relational Databases (*Tobin Bradley, GISP, Mecklenburg County NC*)
PrimaGIS: Collaborative mapping system for Plone CMS (*Chris Calloway, UNC Chapel Hill*)
Web Clients: Community MapBuilder & MapBender (*Julia Harrell, NC DENR*)
Minnesota Mapserver, Chameleon: (*Tom Melhuish, WebInsights*)
GRASS GIS: (*Dr. Helena Mitasova, North Carolina State University*)
Best of the Rest: GDAL/OGR,CartoWeb, GeoServer, MapGuide Open Source (*Doug Newcomb, US Fish & Wildlife Service*)
Mashups: The Non-Profit/Community Perspective (*Wansoo Im, The Center for Community Mapping*)
Southeastern Gap Analysis: Products and Accessibility (*Todd Earnhardt and Jim White, NC State University*)
Econet: NCs Environment and Climate Network for Real-time Weather Modeling (*Ryan Boyles, North Carolina State Climatologist, NC State University*) Note:

All 2007 NC GIS Conference Presentations will be available for download from the NC GIS Conference website[15] within a few weeks.

Julia Harrell
GIS Coordinator, NC DENR
julia.harrell AT Ncmail.net

[15]NC GIS Conference website: http://www.cgia.state.nc.us/ncgis2007

Activity Report of OSGeo-India at Map World Forum 2007

by K. S. Rajan

OSGeo-India actively participated and was well represented at the Map World Forum 2007, held in Hyderabad, during Jan 22-25, 2007. The major activities that the organization organized at this premier international forum of geospatial community were – (i) Keynote address on OSGeo by Gary Lang, Treasurer OSGeo (ii) OSGeo Workshop and (iii) OSGeo Exhibition Booth. In addition, the first meeting of the Executive council of OSGeo-India was held.

OSGeo Workshop

The OSGeo workshop was conducted on 24th Jan, during the afternoon session. The main aim of this workshop was to provide for a platform to disseminate and educate the geospatial community of the enormous opportunities that FOSS4G provides, the advances and the level of technology development in order to provide effective geospatial solutions. During the workshop different models of development and use of FOSS tools and its potential in the domains of academia, government (mainly for organizations mandated with national development, and e-governance), and business were presented. The workshop also had the presence of other countries from Asia – China, Japan and Vietnam.

Prof. Deekshatulu, Ex-Director National Remote Sensing Agency, India, delivered the inaugural address and highlighted the advantages that FOSS tools provide for a large country like India, and in particular emphasized the advantages of the use of FOSS4G tools in the different decision making bodies of the country using these tools with Remote Sensing. He also applauded the timing of this effort and hoped that it will help mobilize the geospatial community in the country to spread the use of these technologies to a much wider public audience.

Mr. Geoff Zeiss of Autodesk shared the changes in the business ideas and how partnering with Open Source is of mutual benefit. He also emphasized the need for concerted efforts towards Open Standards and Open Content in addition to Open Source, which can help create further business opportunities and how it can help in building a Spatial IT ecosystem.

Dr. P.S.Roy, VP OSGeo and OSGeo-India Representative, informed the audience about the efforts in the establishment of OSGeo-India and the plans including outreach, which it has set for itself in the short and mid-term. Some of the ongoing Indian initiatives taken up by organizations like NRSA/DOS, GSI, SACON. NIC, KELTRON, etc. were also shown in the presentation. He also mentioned the challenges and opportunities in the Indian context, especially with the establishment of Village Resources Centers and Krishi Vigyan Kendras, which are mandated to provide a range of services and information including geospatial information. He encouraged all the participants of the workshop to actively participate and contribute in developing a shared vision for the community.

Representatives of China, Japan and Vietnam gave an overview of the efforts of their respective chapters. Prof. Rongguo Chen, VP OSGeo and OSGeo-China Representative, showed the example of the work being carried out at localization, GRASS C-API, and the proposal to setup a GridGIS. While, Mr. Toru Mori, VP OSGeo and OSGeo-China Representative, gave a good example of how open source model is a good business proposition too. Prof. Venkatesh Raghavan, Board Member of OSGeo, in his closing remarks mentioned what the future has in store for the community and the long road ahead.

The workshop had an overwhelming response, with the room overflowing with participants/audience, and was estimated at over 95 persons.

Exhibition Booth

OSGeo-India put up an Exhibition Booth during the Map World Forum, to showcase its efforts: the launch of the India Chapter, disseminate information on OSGeo-India, demonstrate some of the ongoing projects, and provide for a personal interaction between Chapter members and the geospatial community. It had a great response and had a continuous stream of visitors from all sectors. It is worth mentioning that a lot of them commended this effort and our objectives, and shared their views on its relevance to their respective work or organizations. A few of them also invited the Chapter to organize some events like tool demonstration and training in different parts of the country. A sizeable number responded positively and have agreed to continuously interact with OSGeo as users, partners, developers or promoters in their own respective organizations.

OSGeo-India Meeting

The other important event that took place on the sidelines of the Map World Forum was the first meeting of the Executive Council of OSGeo-India. The council had a good discussion on how it would like to take the efforts forward and develop on the response at this first major activity. It has brought on board representatives of academia, government and industry to help make it an inclusive effort and has also consciously provided for a pan-India representation to the Chapter. Representatives of China, Japan and Vietnam attended the meeting as observers, and there was a discussion on how to build interaction and synergy between the different Chapters.

All these efforts of OSGeo-India were made possible by the generous financial support that was extended by the GNOME Foundation and OSGeo.org. OSGeo-India would like to thank these organizations for the support that not only made the logistics of the event possible but also helped provide partial travel support for some delegates from Asia. It is hoped that such efforts will go a long way in fulfilling the objectives of our respective organizations, while looking forward to a continued interaction in the future.

OSGeo-India is now seeking to gain legal status as a non-profit organization in India which would help in promoting the cause and mission of OSGeo in India. Fundraising efforts to support training, software developing and localization has also been initiated. The OSGeo India Chapter is also planning to establish a website to disseminate and propagate the adoption and deployment of Open Source Geospatial solutions in India. Further details about the OSGeo-India Chapter activities can be obtained by contacting the address below.

K. S. Rajan
Treasurer, OSGeo-India
International Institute of Information Technology
Gachibowli, Hyderabad 500032, Andhra Pradesh, India
rajan AT iiit.ac.in

8th Annual Italian GRASS & GFOSS Meeting

by Andrea Scianna

The **VIII edition of the Meeting of Italian GRASS & GFOSS** (*Geographic Free Open Source Software*) users took place on February 14th,15th and 16th in Palermo – Italy, organized by GISLAB at Dipartimento di Rappresentazione of University of Palermo[16]

The meeting registered a strong participation, from all Italian regions, of researchers, students, professionals, and public servants remarking the great interest towards *Geographic Free Open Source Software* in Italy, used in many different sectors (such as environmental monitoring, processing, visualization and publication of geospatial information on workstations or via WEB, etc.)

Even if, in the past, the reference *Free and Open Source GIS* software has been GRASS, today there are many other *GIS* tools available for processing, visualizing and publishing on Internet of geospatial data such as MapServer[17], QGIS[18], PostGIS[19], ka-Map[20], Chameleon[21], PyWPS[22], and more.

The meeting was characterized by a program rich with events.

On February 14th, with the collaboration of Italian Society of Photogrammetry and Topography, the day entitled "Introduction to GIS" took place with

[16]GISLAB at Dipartimento di Rappresentazione of University of Palermo: http://gislab.dirap.unipa.it/
[17]MapServer: http://mapserver.gis.umn.edu/
[18]QGIS: http://qgis.org/
[19]PostGIS: http://postgis.refractions.net/
[20]ka-Map: http://ka-map.maptools.org/
[21]Chameleon: http://chameleon.maptools.org/
[22]PyWPS: http://pywps.wald.intevation.org/index.psp

the participation of professionals and people working in private companies or public administrations that deal every day with surveying and management of geospatial information.

In the same day some tutorials on Linux, and on GIS software (GRASS and QGIS) and on the WebGIS software MapServer were offered.

Topics of the main part of the meeting, held on February 15th and 16th were applications of Geographic Free and Open Source Software, such as geostatistical analysis, GIS techniques for environmental analysis, hydrogeology applications, heritage applications, interoperability, and WebGIS security.

During the meeting also topics of geospatial data standardization were discussed, as well as data exchange problems for public administrations, including intervention of the Italian government.

Besides the interest of public administration to the "Open Source" phenomenon, in Italy there is a strong community of scientists, developers and users that sustain the development of GFOSS. Some of them founded the *Associazione Italiana per l'Informazione Geografica Libera / GFOSS.IT* (Italian association for free geoinformation – GFOSS.it) during the meeting.

The main goals of this new association are sustaining the development, diffusion and rights of Free and Open Source software, promoting open standards for geoinformation and free access to state-collected geospatial data, promoting relationships between people that constitute the Free and Open Source Geospatial Community. *GFOSS.it* intends to become the Italian OSGeo chapter.

The proceedings of the meeting are published on the meeting Web pages[23]

In the near future, the proceedings will also be published on the Web pages of Politecnico di Milano – Polo di Como[24] where papers of all previous Italian GRASS Meeting are published, starting with the year 2000.

Ph.D. Ing. Andrea Scianna
Management board, http://GFOSS.it
CNR D.C.S.P.I. - Dipartimento di Rappresentazione
Università degli Studi di Palermo
Italy
http://gislab.dirap.unipa.it
scianna AT dirap.unipa.it

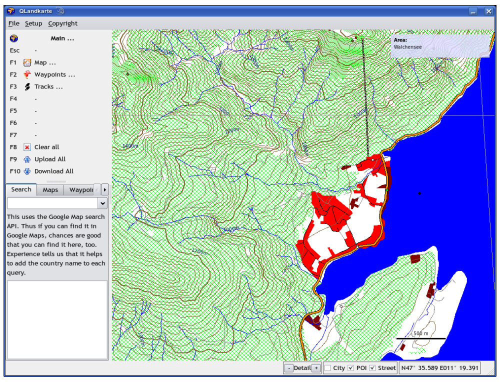
[23]GRASS/GFOSS meeting web page: http://gislab.dirap.unipa.it/grass_meeting/index.htm
[24]Proceedings published at Politecnico di Milano – Polo di Como: http://geomatica.como.polimi.it/workbooks/

Project Spotlights

QLandkarte

by Oliver Eichler

There are many great open source tools like GPS-Babel[25] and GPSMan[26] to manage and manipulate your private geo data, but none of them makes use of Garmin's proprietary vectorized maps. QLandkarte fills in the gap and lets you visualize these maps together with the geo data from your Garmin GPS receiver. It uses Troll Tech's Qt 4.2 library as the GUI toolkit and the Proj4 library for all 2D projections.



This uses the Google Map search API. Thus if you can find it in Google Maps, chances are good that you can find it here, too. Experience tells us that it helps to add the country name to each query.

Figure 1: QLandkarte after start-up

[25] http://www.gpsbabel.org
[26] http://www.ncc.up.pt/gpsman/

Features

- Supported devices:
  - GPSMap 60CS, 60Cx, 60CSx
  - eTrex Legend C
- Import map collections (needs *.tdb file, basemap and map tiles)
- Load single map tiles (*.img Format)
- Display map and object information close to cursor
- Search for places via Google Maps
- Search for geocaches via www.opencaching.de
- Maps:
  - Upload maps to device
  - Download map selection from device
  - Load / save map selection from / to GPX files
- Waypoints:
  - Upload / download waypoints
  - Create / move / edit waypoints
  - Load / save waypoints from / to GPX files
- Tracks:
  - Download tracks

  - View track information and elevation profile
  - Combine / split tracks
  - Purge / remove track points
  - Load / save tracks from / to GPX files
- Print maps to PDF or paper

Overview

Figure 1 shows QLandkarte right after program start. Function and cursor modes can be switched by keyboard or mouse selection in the top left menu. User data will be listed in the tabs below.

There are three main function modes: maps, waypoints and tracks. The key assignment in each of these modes is displayed in Figure 2. Keys F1 to F4 have static functions assigned that let you navigate your map. A dynamic zoom is done by scrolling the mouse wheel.

Each main mode has its own dialog in the tab widget, listing the available user data. Selected data can be removed with the DEL key as a general rule.

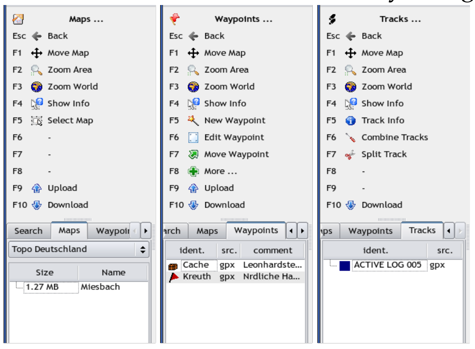


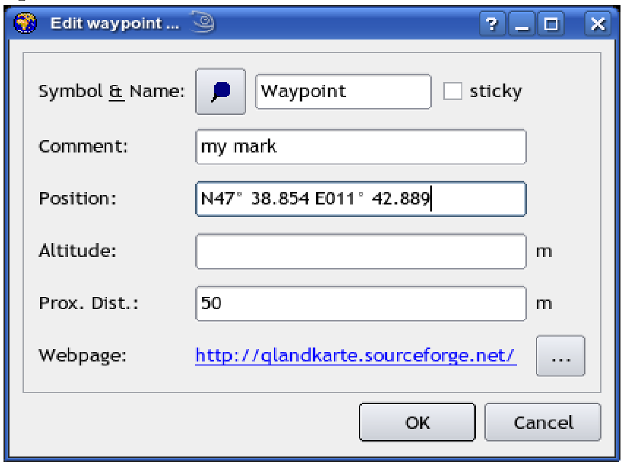
Figure 2: Function Modes

Map Mode

In 'Map Mode' the current map collection can be selected from the combo box at the tab widget's top. To upload map tiles to the device they have to be selected by point-and-click in the map view area. Selected tiles from all collections are listed in the tree widget.

Waypoint Mode

Figure 3 presents the waypoint properties dialog. Next to the usual fields like symbol, name and position, a distance for proximity alarms and a link to a web page can be given. The link is active and will open a browser.


Edit waypoint ...

Symbol & Name:	Waypoint ☐ sticky
Comment:	my mark
Position:	N47° 38.854 E011° 42.889
Altitude:	m
Prox. Dist.:	50 m
Webpage:	http://qlandkarte.sourceforge.net/ ...

OK Cancel

Figure 3: Waypoint Properties

A double click in the waypoint list will center the map around the selected waypoint. The 'Move Waypoint' mode is quite complex and should get special attention. A left button click on a waypoint will glue it to the cursor. Additionally a rubber band will span between the initial position and the current one, displaying the distance together with the forward and backward azimuths. See Figure 4. In this state it is possible to switch the mode to the map navigation functions, allowing the user to move the map to the desired destination area. Once back in the 'Move Waypoint' mode the waypoint can be dropped by a second left button click or the operation is escaped by a right button click.

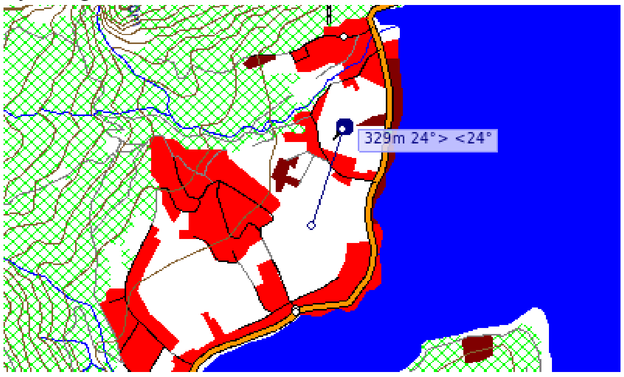
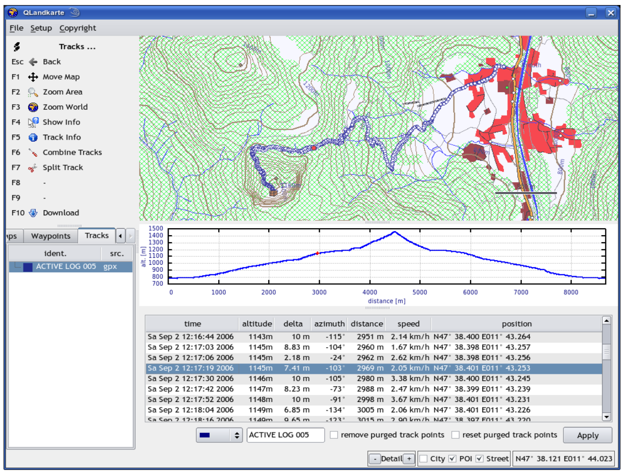


Figure 4: Moving a Waypoint

Track Mode

In Figure 5 QLandkarte operates in 'Track Mode' . The track info view has been raised. A double click in the track list will zoom the map view to fit the selected track. The track is displayed as bread crumbs on the map, as elevation profile and as table. All three views are linked together, thus if a track point is selected in one of them, it will get highlighted in the other ones, too. The visibility of track points can be toggled with the DEL key in the table view. This will gray shade the points in the table and prevent them from adding to the the track.

Multiple selected tracks can be combined into a new one. The current selected track can be split at any point. QLandkarte will not destroy the original data. It will create two new tracks as copy of the original one.



time	altitude	delta	azimuth	distance	speed	position
Sa Sep 2 12:16:44 2006	1143m	10 m	-115°	2951 m	2.14 km/h	N47° 38.400 E011° 43.264
Sa Sep 2 12:17:03 2006	1145m	8.83 m	-104°	2960 m	1.67 km/h	N47° 38.398 E011° 43.257
Sa Sep 2 12:17:06 2006	1145m	2.18 m	-24°	2962 m	2.62 km/h	N47° 38.398 E011° 43.256
Sa Sep 2 12:17:19 2006	1145m	7.41 m	-103°	2969 m	2.05 km/h	N47° 38.401 E011° 43.253
Sa Sep 2 12:17:30 2006	1146m	10 m	-105°	2980 m	3.38 km/h	N47° 38.400 E011° 43.245
Sa Sep 2 12:17:42 2006	1147m	8.23 m	-73°	2988 m	2.47 km/h	N47° 38.399 E011° 43.239
Sa Sep 2 12:17:52 2006	1148m	10 m	-91°	2998 m	3.67 km/h	N47° 38.401 E011° 43.231
Sa Sep 2 12:18:04 2006	1149m	6.85 m	-134°	3005 m	2.06 km/h	N47° 38.401 E011° 43.226
Sa Sep 2 12:18:16 2006	1149m	9.65 m	-123°	3015 m	2.90 km/h	N47° 38.397 E011° 43.220

Figure 5: Track Info View

Development

The project has reached a stage satisfying the original itches of the author. This does not necessarily imply that it is a mature and full featured application.

QLandkarte still misses support for many Garmin devices. It provides a plug-in system for devices to allow a decentralized development by several unit owners. Writing a driver is easier than it sounds. There is a small tutorial on QLandkarte's homepage.

Another vacant task would be porting QLandkarte to other platforms like Windows or Mac. It shouldn't be too much of a problem for any little endian platform. However there has to be much work done for big endian platforms as the IMG data is stored in little endian format and the decoder addresses the data directly for performance sake. Thus a big endian port should compensate the performance penalty for byte shifting by optimizing the decoder code.

Interested developers are welcome to discuss all this and send their patches on the QLandkarte's user mailing list.[27]

Dipl.-Ing. (FH) Oliver Eichler
The author studied Electronic Engineering
in Regensburg and since 1997 has worked
as a freelancer mainly focused on digital subscriber lines.
http://qlandkarte.sourceforge.net
oliver.eichler AT gmx.de

A Short Introduction to Mapbender

"...bend the maps to suit your needs"

by Astrid Emde and Marc Jansen

Introduction

This article gives an overview of the scope, character and functionality of the web mapping client software Mapbender.[28] A brief introduction to the functional scope of Mapbender is followed by the history of the project. Then the main functionality is described from both the management and orchestration as well as the visualization and presentation perspective of web mapping and feature services. The final section summarizes future development in a roadmap for the next year. Two productive environments are briefly presented and links to more examples give an overview of what Mapbender is being used for.

Mapbender
Geoportal Framework

What is Mapbender?

The Mapbender Client Suite is a framework to manage, orchestrate and display web services for spatial data supporting the OGC standardized WMS and

[27]http://sourceforge.net/mail/?group_id=183501
[28]See the Developer Announcements section on page 57 for more information about Mapbender

WFS interfaces. Mapbender is not a map server software like the University of Minnesota MapServer[29] but is used to display, overlay and modify OGC maps rendered by MapServer and any other OGC compatible server software.

In Mapbender, maps from various sources are stacked on top of each other and can then be distributed throughout the world via the Internet. As the OSGeo site states, *Mapbender is the tip of the iceberg, a meta layer of software providing access to the Open Source Geospatial SDI (Spatial Data Infrastructure) stack.*[30]

The software provides interfaces for displaying, navigating and querying OGC OWS services, e.g. WMS (Web Map Service), WFS (Web Feature Service), WFS-T (transactional WFS) and WMC (Web Map Context). The Mapbender framework additionally provides interfaces for user and group administration. Management functionality for access to maps rendered and data served by OGC Open Web Services is included as well. The Mapbender database model has been designed to implement multi-client capabilities with respect to users, groups, interfaces and services. The following three basic components make up Mapbender:

1. Users and Groups
2. User Interfaces (graphic user interface, GUI)
3. Map Services (OGC Open Web Services)

Currently the main focus of development has shifted to WFS-T functionality, catalog and gazetteer services.

Short history of Mapbender

At the end of the last millennium the Germany-based company CCGIS created the first PHP-based web map interface to display OGC web map services (then only available as a WMT discussion paper). This software developed into a product and eventually became the basis of the current Mapbender. In November 2001 the first version of the 'CCGIS Client Suite' was published implementing the first OGC WMS standard.

In 2002 the first larger installation of this proprietary software package was set up as a productive environment serving 400 users. The year 2003 saw several important changes in the project:

- The suite was completely redesigned with two goals in mind:
  1. Reduce the complexity of the relationship of services, interfaces, interface elements and user management.
  2. Provide administration interfaces and a data model allowing users to manage every aspect of the software via web interfaces. The administration interfaces are managed recursively by themselves vastly reducing the complexity of the software.
- Users, developers and licensees of the suite were asked for their opinion on a major change of philosophy, as CCGIS planned to change to a Free Software license.
- Licensees were happy to see this happen and so the software was released under the Free Software license GNU GPL.
- The new name of the project was announced: The 'CCGIS Client Suite' was henceforth to be called 'Mapbender'.

In October 2003 more than 500 downloads were counted within days of the last release update. Users were accepting the new license and starting to adopt the new open development philosophy. One year later, in 2004, the software was being used productively in five public administrations (counties and cities) that are part of the Spatial Data Infrastructures of North Rhine-Westphalia.[31]

In 2004, an official demo server was put into operation hosted on CCGIS hardware. It is currently in transition to a new server[32] which will then be the only official server. Additionally, in 2004, the website was merged with the documentation into a Wiki as common technical platform. Later this Wiki was copied to MediaWiki software which provided more functionality and badly needed user rights management to overcome spam issues.

April 2005 saw the first user conference which took place in Bonn, Germany and attracted 85 attendees from Germany, Austria, Switzerland and the Netherlands. It was also the last exclusive Mapbender user meeting, since then it has been combined

[29] University of Minnesota MapServer: http://mapserver.gis.umn.edu/
[30] From http://www.osgeo.org/mapbender
[31] Spatial Data Infrastructures of North Rhine-Westphalia: http://www.gdi-nrw.org
[32] New Mapbender demo server: http://mapbender.telascience.org
[33] FOSSGIS: http://www.fossgis.de
[34] FOSS4G 2006: http://www.foss4g2006.org

with existing conferences like FOSSGIS[33] or FOSS4G 2006.[34]

Early in 2006, Mapbender became one of the eight initial projects of the Open Source Geospatial Foundation (OSGeo). In the following 6 months Mapbender went through the incubation process of the OSGeo under scrutiny of the Incubation Committee and the help of project mentor Paul Spencer. During incubation several license issues were resolved, the code was cleaned, copyright ambiguities resolved, source code was moved from SourceForge CVS to CollabNet SVN as well as mailing lists and parts of the infrastructure. On July 19th 2006 Mapbender was the first project to officially graduate from incubation.

Yet another infrastructure move was due at the end of 2006 when the project infrastructure moved to OSGeo's own hardware and a more free and open infrastructure. Finally (almost) all components of the development environment and communication infrastructure are managed under one roof, the only component still running on a separate server being the Wiki.

The current stable version of the software is 2.4.1, which was released on March 12th 2007.

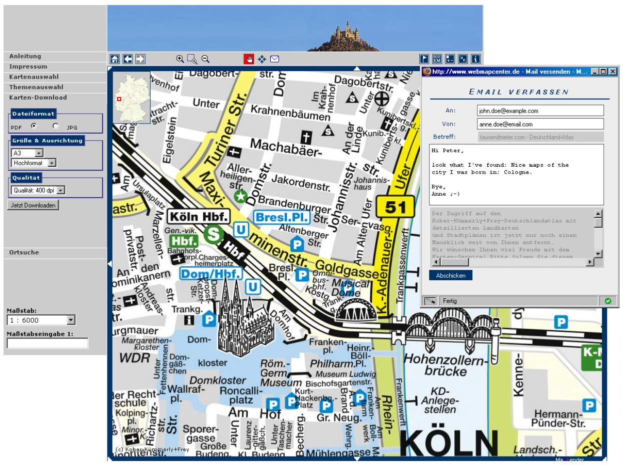


Figure 1: A demo application of Mapbender. Notice the toolbar (top), Menus (left, visible: PDF-options) and the email-dialog to send a link of the current extent (upper right corner). Image copyright: Kober-Kï£¡mmerly+Frey Media AG, Cologne, Germany. Used by permission.

Functionality

Mapbender provides functionality to manage and display spatial data rendered and served by OGC standardized software. Find examples for different scenarios in the Mapbender Gallery[35] or go to the official online demo application.[36] The standard interface demo and a full fledged geoportal are currently in transition to OSGeo servers hosted by Telascience.[37]

Manage Web Mapping Services

Mapbender is used to orchestrate OGC services and formats. The software provides administration interfaces to upload and cache web service capabilities documents for WMS, WFS and transactional WFS. Once the configuration description is uploaded it can be combined with map overlays and features from other services and displayed through map interfaces.

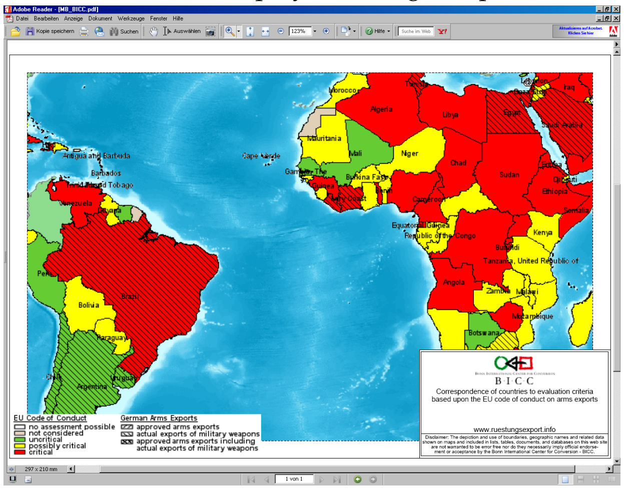


Figure 2: A DIN A4 landscape PDF that was generated on-the-fly by Mapbender. It shows actual German exports of weapons (hatching) and the "EU Code of Conduct for Human Rights" (colored) for parts of South-America and Africa. See http://www.ruestungsexport.info.

As many web services lack metadata information Mapbender has been extended with an editor that allows users to broaden existing information and map the service and each layer to categories as specified in Dublin Core.[38] This enhanced information can

[35]Mapbender Gallery: http://www.mapbender.org/index.php/Gallery
[36]Online demo application: http://www.mapbender.org/index.php/Demo
[37]New Mapbender demo server: http://mapbender.telascience.org
[38]Dublin Core http://dublincore.org/

be queried and searched with the metadata search module and can be displayed by following a link that is attached to the TreeGDE (Geo Data Explorer). The TreeGDE is a directory structure used to manage layers and services in the display interface.

Manage Users, Groups and Services

A comprehensive fully client capable user management allows to select and configure interfaces with WMS and WFS collections for users and groups. The Mapbender OWS Proxy module can be activated to be used as a facade in front of services. Ticketing has been implemented using Apache redirect functionality providing for an interface that is at the same time OGC compliant but also allows for restricted access. This can also be used to implement protocol encryption which in some cases may require the use of a client side proxy facade as implemented in the InteProxy software (http://inteproxy.wald.intevation.org/). All requests are then routed through the user management module and permission can be denied for requests that query services which the user is not allowed to access.

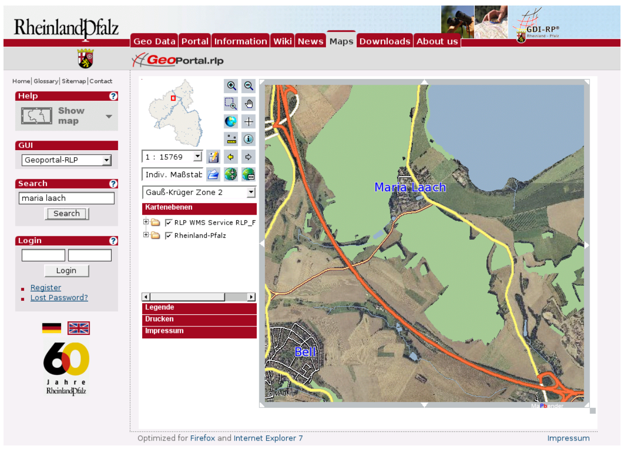


Figure 3: The Geoportal of Rhineland Palatinate (http://www.geoportal.rlp.de/) gives access to more than 70 WMS map services with several hundred layers operated by the public administration, non-government institutions and private enterprises. It enables service operators to upload and manage their own services, edit corresponding meta data and allow users and groups to access them.

Visualize, Navigate and Query Services

Besides the standard functionality like pan, zoom, query it is possible to digitize (create, edit, delete) geometries (features) and store them using WFS-T interfaces. WFS is also used to search, find and navigate to features (street address, land parcel, moving vehicle with GPS online coordinate transfer) and can perform operations.

OGC Web Map Context (WMC) documents can be created, stored and retrieved to return to a defined state of the web interface.

All functionality in Mapbender is implemented as modules that can be combined into user interfaces with the web based administration.

A full alphabetical list of implemented GIS functionality can be found online.[39]

Future development

Mapbender is dynamically extended and improved through input from the community. Future releases will contain additional functionality and solve bugs and issues:

- Enhanced logging of user activity,
- Print module with more user friendly configuration,
- Print (and PDF) output to support temporary created geographic overlay objects,
- Include catalog services interface extension to connect with GeoNetwork opensource,
- Enhanced tree folder editing,
- Online routing geometry,
- Modularized multilevel gazetteer services,
- Many more things that we don't even dream about yet.

Further down the road, an integrated SLD (Styled Layer Descriptor) editor is needed to help customize cartographic output.

All input by users is highly appreciated, subscribe to the mailing lists[40] and share your ideas and thoughts.

References and weblinks

- Project Wiki and Homepage (primary source of information used in this article): http://www.mapbender.org/index.php/Main_Page

[39]GIS functionality list: http://www.mapbender.org/index.php/Category:Modules
[40]Mapbender mailing lists: http://www.mapbender.org/index.php/Mapbender_Mailing_Lists

- Documentation, first steps and general help on various topics: http://www.mapbender.org/index.php/User_Documentation
- Project page located on the Open Source Geospatial Foundation's website: http://www.osgeo.org/mapbender
- Screenshots, links & examples of various Mapbender installations: http://www.mapbender.org/index.php/Mapbender_Gallery
- Online demo server: http://www.mapbender.org/index.php/Demo
- Mailinglists, SVN code repository and Internet Relay Chat (IRC): http://www.mapbender.org/index.php/Mapbender_Mailing_Lists
- The logo is available online at: http://www.mapbender.org/images/2/21/Mapbender_Geoportal_Framework.png

Astrid Emde
WhereGroup GmbH & Co.KG
http://www.wheregroup.com/
astrid.emde AT wheregroup.com

Marc Jansen
terrestris GbR
http://www.terrestris.de/
jansen AT terrestris.de

deegree – Building Blocks for Spatial Data Infrastructures

by Markus Müller

Introduction

deegree[41] is a Java-based Open Source / Free Software framework for the implementation of Spatial Data Infrastructures (SDI). It contains the services needed for SDI (deegree Web Services) as well as portal components (deegree iGeoPortal), mechanisms for handling security and access control issues (deegree iGeoSecurity) and storage / visualization of 3D geodata (deegree iGeo3D). deegree is conceptually and interface-wise based on the standards of the Open Geospatial Consortium (OGC) and ISO / TC 211. At the time of writing it is the most comprehensive implementation of those standards in one Open Source framework. The framework is component-based to a high degree, allowing the flexible creation of solutions for a wide variety of use cases. deegree is the official reference implementation of the OGC for the Web Map Service and Web Coverage Service standards. It is published under the GNU Lesser General Public License (LGPL).

Historical background

deegree is managed in a cooperation between the private company lat/lon and the GIS working group of the University of Bonn. The roots of deegree go back to a project of the University of Bonn named EXSE (GIS EXperimental SErver) running from 1997 through 2003. The aim of the project was an experiment-based analysis of merging GIS functionality and internet technology. Several tools and software modules had been developed including a first implementation of the OGC Simple Feature for CORBA specification as an Open Source Java API (called sf4j - Simple Features for Java).

In spring 2001, the sf4j project, the existing tools and software modules were re-arranged into a new project called JaGo (Java Framework for Geospatial Solutions) aiming to realize an Open Source implementation of the OGC Web Service specifications. The first service implemented was the OGC WMS 1.0.0 specification in summer 2001. Until the end of that year WFS 1.0.0 and WCS 0.7 followed. As the important web domains (.de, .org, .net) for JaGo were not available it was decided at the end of 2001 to rename the project to deegree.

The next important step was the release of deegree 1.0 in late summer 2002. Some changes in the architecture offered a better handling of the available OWS (OGC Web Services). An implementation of a multi-threading service engine and interfaces to remote OWS enabling high scalability of applications were added. The following minor versions featured new functions like a transactional WFS, a Gazetteer (WFS-G) and support of additional data sources. From this time on, deegree WMS supported

[41]degree website: http://www.deegree.org

Styled Layer Descriptor (SLD) and a Catalogue Service. Security mechanisms were added to the framework. An important step for increasing the publicity of the project was moving it to sourceforge as its distribution platform. Several developers started reviewing the deegree code base and adding code to the project.

An additional working thread in the development of the framework has been started in 2003. It aims at offering components to enable developers to create web clients based on deegree. This new client framework is named iGeoPortal supports the OGC standard Web Map Context 1.0.0.

One of the most important steps in deegree development was participation in the OGC CITE1 project in summer 2003, that resulted in deegree becoming the official OGC reference implementation for the WMS 1.1.1 specification. The participation of lat/lon and deegree in CITE was so successful that lat/lon has been selected by the OGC later on to develop WCS 1.0.0 and WMS 1.3 reference implementations with deegree in context of OGC's OWS-2 and OWS-4 initiatives.

In 2005, deegree2 was launched, representing again a great step forward in the development of the framework. The keystones of deegree2 are a model-based mechanism for deegree WFS, allowing flexible implementation of different data models using GML application schemas. Additionally, the development of a portlet-based client framework called deegree iGeoPortal - portlet edition, support for 3D data structures and a Web Processing Service (WPS) implementation are included in deegree2.

Components

deegree comprises five groups of components that are briefly described in the following:

deegree Web Services

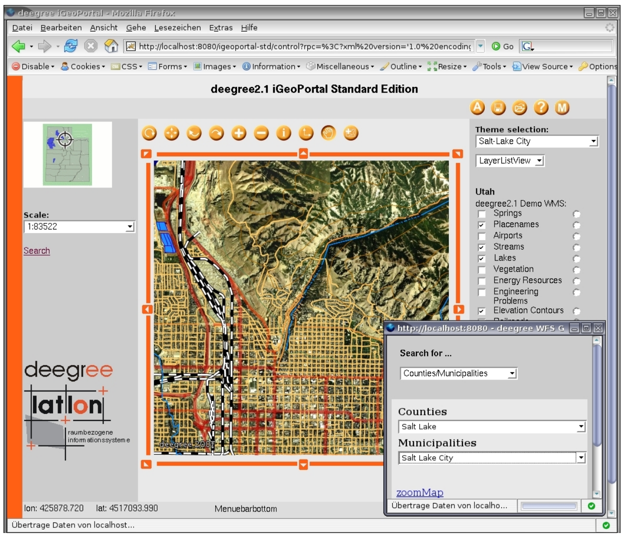
These are the base components of any SDI. They can for example be used to display maps and 3D data, access vector and raster data, metadata management and retrieval, and processing of geodata. The list of implemented OGC Web Services includes WMS, WFS, WCS, CS-W, Gazetteer, SOS, WPS and WTS/WPVS. Besides, two security services (WAS/WSS), a printing service (WMPS) and a service monitor (owsWatch) are available.

deegree iGeoPortal

This is the portal framework of the deegree project. iGeoPortal has a modular structure and is able to display maps, support searches using geographic identifiers, search for datasets using metadata, allow for controlled access to OGC Web Services and display 3D geodata.

Different kinds of SDI portals are created using deegree iGeoPortal standard and portlet edition and the corresponding geo web services. While deegree standard edition is using DHTML technology, the portlet edition is based on JSR-168, the portlet standard. Both editions use AJAX technology for some specific modules. These portals include standard WebGIS and specialized applications as used by municipalities, environmental, surveying and other agencies.

deegree iGeoPortal itself consists of different modules for separate, but combinable, functionalities. The list of modules includes: map, download, gazetteer, catalogue, security and 3D.


deegree iGeoPortal - Mozilla Firefox

http://localhost:8080/igeoportal-std/control?rpc=%3C?xml%20version='1.0%20encoding

deegree2.1 iGeoPortal Standard Edition

Scale: 1:83522

Theme selection: Salt-Lake City

LayerListView

Utah
deegree2.1 Demo WMS:
- [ ] Springs
- [x] Placenames
- [ ] Airports
- [x] Streams
- [x] Lakes
- [ ] Vegetation
- [ ] Energy Resources
- [ ] Engineering Problems
- [x] Elevation Contours

http://localhost:8080 - deegree WFS G

Search for ... Counties/Municipalities

Counties: Salt Lake

Municipalities: Salt Lake City

lon: 425878.720 lat: 4517093.990 Menuebarbottom

Übertrage Daten von localhost...

Figure 1: deegree iGeoPortal standard edition.

deegree iGeoSecurity

Access control for geodata and services is an important issue for a wide variety of applications. deegree iGeoSecurity can be used to define access mechanisms using authentication and authorization mechanisms, secure connections and filtering of geodata. A database for managing users, user groups, roles and rights called deegree U3R is the core component of the security components.

deegree iGeo3D

deegree can be used to store 3D geodata such as digital terrain and building models in file-based systems and relational databases. Using different deegree web services, this data can be queried and displayed. These systems are often used by surveying agencies and municipalities to store their 3D data for commercial purposes or to fulfil EU regulations such as the European noise guidelines. deegree iGeo3D uses the OGC standards WFS, WCS, WTS and CityGML.

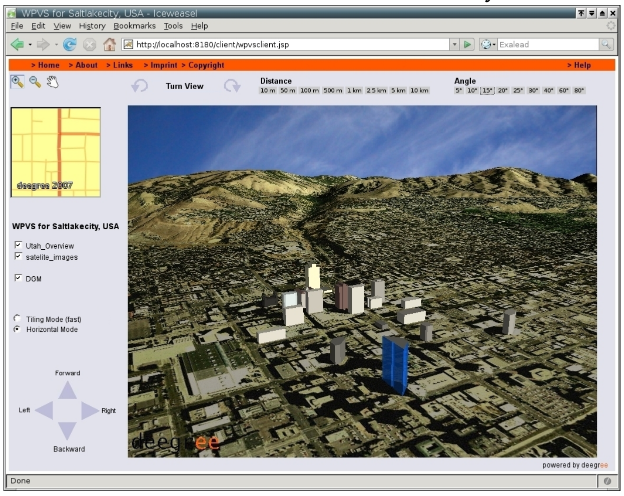

WPVS for Saltlakecity, USA - Iceweasel
File Edit View History Bookmarks Tools Help
http://localhost:8180/client/wpvsclient.jsp
> Home > About > Links > Imprint > Copyright
> Help
Turn View
Distance
Angle
WPVS for Saltlakecity, USA
Utah_Overview
satelite_images
DGM
Tiling Mode (fast)
Horizontal Mode
Forward
Left
Right
Backward
powered by deegree
Done

Figure 2: deegree iGeo3D.

deeJUMP

An additional application area of deegree is the classical desktop GIS. Using deegree components, the Open Source desktop GIS JUMP (Java Unified Mapping Platform) was enhanced to support WMS and WFS. A number of additional modifications and extensions were also developed.

Overall architecture

The different deegree components can be combined with each other or with other standard-compliant software. An exemplar architecture of a deegree-based system is displayed in figure 3.

iGeoPortal with its different modules acts as a web-based client to the different Web Services. Alternatively, deeJUMP can be used to access WMS or WFS services. If needed, security management can be implemented using iGeoSecurity. deegree can access different data sources, e.g. PostgreSQL/PostGIS or Oracle databases, shapefiles, all kinds of relational databases using a deegree-specific spatial extension and different kinds of image formats such as PNG, GIF, JPEG or (Geo)TIFF.

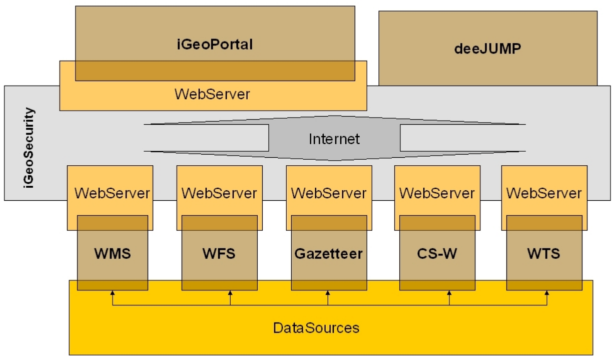


Figure 3: deegree architecture.

Current status

In June 2007 deegree day will take place in Bonn Germany. Until then, the 2.1 releases of deegree will be published. The corresponding demo releases will be extremely easy to install as they come as WAR-archives and can be deployed within seconds if a Tomcat Servlet engine is available. The demo releases include a Web Map Service, a Web Feature Services (with gazetteer), a Web Coverage Service, a Catalogue Service-Web (for ISO metadata), a Web Terrain Service (Web Perspective View Service), iGeoPortal and deeJUMP. All demo releases come with preconfigured data sources and detailed documentation. The 2.1 releases are a major step forward in deegree's development. Feel free to visit `http://www.deegree.org` for news regarding deegree day 2007 and the upcoming releases.

Dr. Markus Müller
lat/lon GmbH
Aennchenstr 19
53177 Bonn
Germany
`mueller AT lat-lon.de`
`http://www.lat-lon.de`

Introducing openModeller

A fundamental niche modelling framework

by Tim Sutton, Renato de Giovanni and Marinez Ferreira de Siqueira

Introduction

In 1957 Hutchinson (4) formalised the fundamental niche principle. In essence, he proposed that if all the environmental conditions that allow a species to exist indefinately could be tabulated as a multidimensional hypercube, then the resulting hypercube could be considered to be the organism's fundamental ecological niche. Naturally, fully understanding all the ecological conditions for any given species is a monumental task, limited primarily by the lack of rich data that could drive such an analysis. By extrapolation of the association between species occurrence localities and a set of GIS raster layers representing environmental conditions such as rainfall, temperature, solar radiation etc., a correlative approach can be taken towards describing an organism's ecological niche (10). Using this technique, a raster layer can be produced with areas that most resemble the environmental conditions at the original known sites of occurrence for the organism. Arguably, this approach first garnered widespread interest with the publication of 'A biogeographic analysis of Australian Elapid snakes.' (5), where the authors produced predicted distribution maps for Elapid snakes based on simple models of climatic preference. As the world starts to engage more fully in the discussion of the potential impacts of global climate change, the abillity to predict the impact of these changes on the distribution of organisms has become more topical. Fundamental niche modelling has been used to predict species loss in the face of future climate changes (12; 2). It has also been used to model invasive species, spread of disease vectors (6) and in phylogenetic modelling to produce distribution maps for hypothetical ancestors (13; 14).

A number of software applications exist for carrying out fundamental niche analysis. Popular choices include Bioclim (5), DesktopGARP (9), MaxEnt (7) etc. Elith *et al.* (3) provide an excellent overview and comparison of some of the more comonly used applications. These applications are usually each created to address a specific algorithmic approach to fundamental niche modelling. Typically these tools are not free (as in freedom), open source software. In addition, each application has different requirements in terms of data preparation, system requirements (Java, MS Windows etc.) and the learning curve required from the user to gain the neccessary skills to use the application. In other cases, algorithms for fundamental niche modelling were developed within scripting toolkits, such as the Climate Space Model (8) which was developed using MatLab. These are difficult to redistribute as user friendly applications.

To address these (and other) concerns, the *openModeller* project was initiated in 2003 by CRIA[42]. The *openModeller* project is free and open source, and all code is available on SourceForge. *openModeller* development is supported by FAPESP[43], and by contributions from the open source developer community. The principle aim of the *openModeller* project is to provide tools for researchers interested in fundamental niche modelling. It is a fresh approach to the world of fundamental niche modelling since it provides a plugin architecture for new algorithms to be incorporated easily. Providing all algorithms under a common architecture allows for easier comparison of results between different algorithms since the all models can be prepared using the same independent data and can be output into a common format. Implementations of GARP, Bioclim, CSM, Environmental Distance and others have already been developed as plugins for *openModeller*.

openModeller capitalises on existing open source software, harnassing libraries such as GDAL[44], GSL[45], QGIS[46] etc. *openModeller* is written in C++ and is cross-platform, running on MS Windows, Mac OS X and GNU/Linux.

openModeller sub projects

There are a number of development areas within the *openModeller* project. First and foremost is the *openModeller* library. The library provides a common plat-

[42]Centro de Referẽncia em Informação Ambiental: http://www.cria.org.br
[43]http://www.fapesp.br
[44]GDAL - Geospatial Data Abstraction Library: http://www.gdal.org
[45]GSL - GNU Scientific Library: http://www.gnu.org/software/gsl/
[46]QGIS - Quantum GIS: http://qgis.org

form for all other tools mentioned below. It includes the ability to read environmental and species occurrence data, setting model parameters, loading and running algorithm plugins and writing the results of an analysis to a geospatial dataset. C++ and Python bindings (generated using SWIG) for the library are available.

A Web Services API based on SOAP (using the document / literal style) is available for remote execution of *openModeller* jobs. A CGI application is available, allowing this service to be deployed inside of an Apache web server instance.

A number of console tools have been created on top of this library. *om_console* provides a way to run a model using simple text configuration files that specify which input datasets to use. *om_create* and *om_project* provide a means to use *openModeller* using XML configuration files. Other helper tools are also available for the command line to provide simple data extraction and metadata query facilities. For example *om_sampledump* extracts the environmental values at each occurrence point and provides the result in a delimited format.

For end users there is a cross platform graphical user interface 'desktop' application (written in C++ and Qt4). *openModeller Desktop* provides a user friendly environment for preparing, running and visualising the results of fundamental niche models. *openModeller Desktop* will be described in more detail in the section that follows.

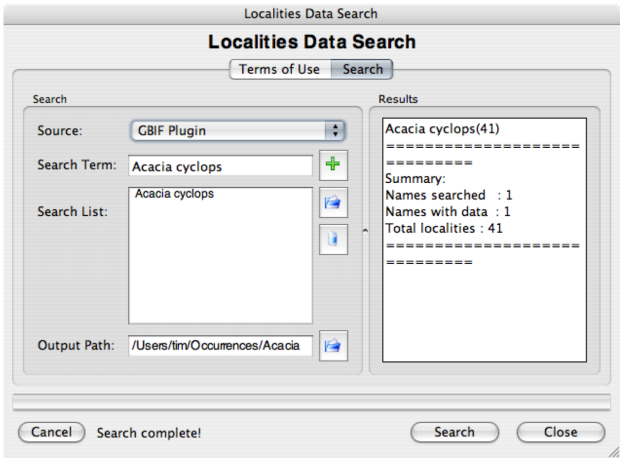
openModeller Desktop

openModeller Desktop is a solution for users wishing to use the *openModeller* library without programming or needing to be familiar with the various command line tools that the *openModeller* project provides. Within *openModeller Desktop*, users can carry out the following core activities:

- automatically retrieve occurrence data from online databases such as GBIF[47] and speciesLink[48];
- manage the environmental raster layers that will be used for modelling;
- manage the parameters for the various algorithm plugins included with *openModeller*;
- design an experiment which will run multiple models for one or more species using one or more algorithms;
- visualise the results of an experiment by viewing individual models using a tabular and a map view.

Modelling Acacia cyclops - a simple case study

One typical use of fundamental niche modelling is to predict the areas that may be vulnerable to the spread of invasive species. In this example *openModeller* will be used to find areas of potential distribution for *Acacia cyclops*[49]. *Acacia cyclops* is a native plant of south western Australia, but is known to be an invasive pest plant in other parts of the world. Thus the aim of this case study is to ascertain areas potentially vulnerable to invasion from this species. To start, the *openModeller Desktop* localities search tool was used to search for locality records (see Figure 1).



Source: GBIF Plugin
Search Term: Acacia cyclops
Search List: Acacia cyclops
Output Path: /Users/tim/Occurrences/Acacia

```
Acacia cyclops(41)
=============================
Summary:
Names searched   : 1
Names with data  : 1
Total localities : 41
=============================
```

Search complete!

Figure 1: Searching GBIF for locality data.

Since the search results returned records outside of Australia, a text editor was used to delete all occurrence records that were not within Australia. This was done in order to model a fundamental niche based solely on the species' known native distribution. This reduced the initial 41 records retrieved from GBIF to 31 records, all of which originate from the Australian Virtual Herbarium (1). After this, the

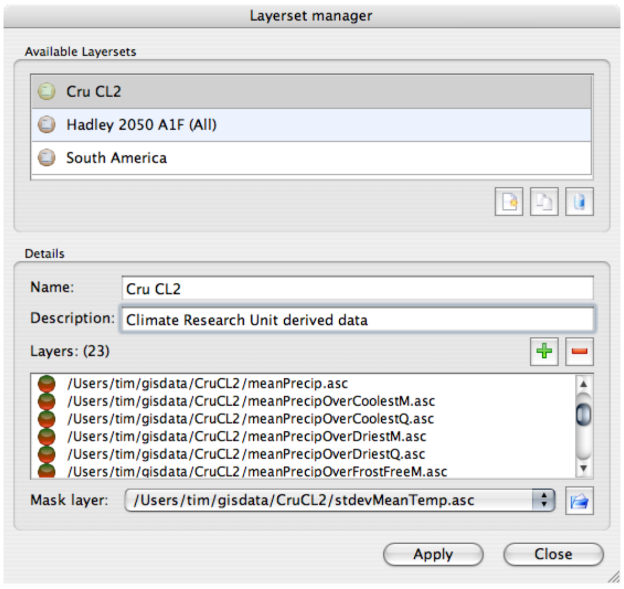
[47]GBIF - Global Biodiversity Information Facility: http://gbif.org

[48]http://splink.cria.org.br

[49]*openModeller Desktop* is capable of modelling many species in a single experiment, but for the purposes of a simple case study only one species was used

[50]CRU - Climate Research Unit: http://www.cru.uea.ac.uk

layerset manager in *openModeller Desktop* was used to define a layerset. 23 layers derived from CRU[50] were selected for this purpose (see Figure 2.). These layers represent climate related indicators (based on recorded climate measurements) related to temperature, rainfall etc.


Layerset manager

Available Layersets

- Cru CL2
- Hadley 2050 A1F (All)
- South America

Details

Name: Cru CL2

Description: Climate Research Unit derived data

Layers: (23)

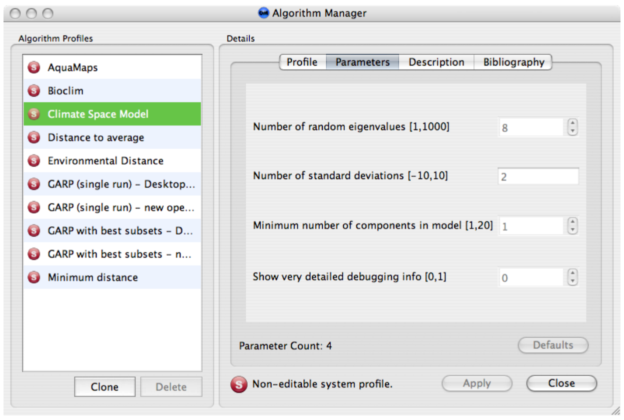
```
/Users/tim/gisdata/CruCL2/meanPrecip.asc
/Users/tim/gisdata/CruCL2/meanPrecipOverCoolestM.asc
/Users/tim/gisdata/CruCL2/meanPrecipOverCoolestQ.asc
/Users/tim/gisdata/CruCL2/meanPrecipOverDriestM.asc
/Users/tim/gisdata/CruCL2/meanPrecipOverDriestQ.asc
/Users/tim/gisdata/CruCL2/meanPrecipOverFrostFreeM.asc
```

Mask layer: /Users/tim/gisdata/CruCL2/stdevMeanTemp.asc

Figure 2: The layerset manager.

Creating named layersets provides great benefit to the user since they can easily refer to multiple layers as one conceptual entity and do not need to spend a lot of time selecting files from the file system each time they run an experiment. Once a layerset has been defined it can be used repeatedly in different experiments.

openModeller Desktop includes an algorithm manager (fig. 3). This tool lets the user view all the available modelling algorithm plugins that have been installed along with the *openModeller* library. All algorithms ship with sensible defaults (as determined by algorithm authors) so that they can be used 'out the box'. Within the algorithm manager these default parameters are presented as immutable 'system profiles'. System profiles can be cloned and modified to create 'user profiles'. Modification of these parameters requires some knowledge of the principles behind the underlying algorithm, but provides advanced users with a great deal of flexibility and control. Novice users can eschew the algorithm manager completely and simply work with the default 'system profiles'.


Algorithm Manager

Algorithm Profiles

- AquaMaps
- Bioclim
- Climate Space Model
- Distance to average
- Environmental Distance
- GARP (single run) – Desktop...
- GARP (single run) – new ope...
- GARP with best subsets – D...
- GARP with best subsets – n...
- Minimum distance

Details

Number of random eigenvalues [1,1000]: 8

Number of standard deviations [-10,10]: 2

Minimum number of components in model [1,20]: 1

Show very detailed debugging info [0,1]: 0

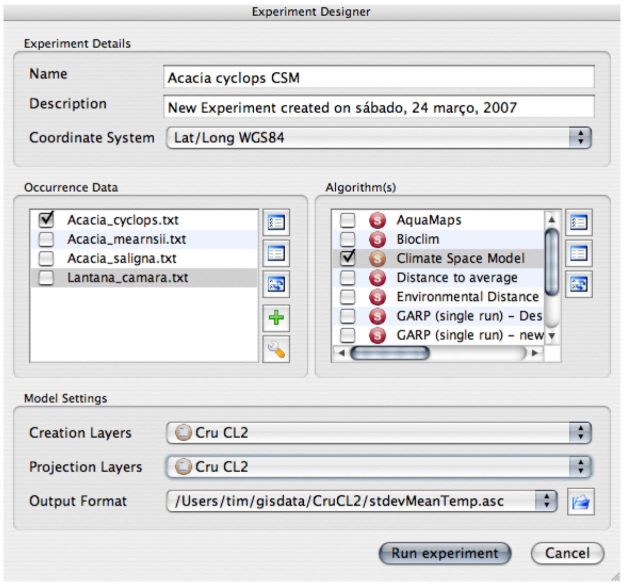
Parameter Count: 4

Non-editable system profile.

Figure 3: The algorithm manager.

The occurrence data, layerset and algorithm profile are the three essential elements for designing a modelling experiment. The experiment designer (see Figure 4.) is used to define how these data should be combined to form an experiment. Each created experiment is given a unique name and a description. The experiment designer provides a simple user interface with powerful functionality. With it, large experiments can be created that model multiple species using multiple algorithms. In the future, support will be added for modelling with multiple climate scenarios. In *openModeller Desktop*, the actual invocation of modelling processing, happens via a 'modelling plugin'. This means that the interaction between the Desktop interface and the *openModeller* library is very well separated. It also means that new modelling plugins can be written to support other modelling toolkits in the future. *openModeller Desktop* currently has two such plugins available: a 'local' plugin which uses libopenModeller directly, and a 'web services' plugin that can invoke modelling jobs on a remote server using the *openModeller* Web Services API. The latter is still somewhat experimental, but the aim is to allow the user to design large, CPU intensive experiments within the user friendly *openModeller Desktop* interface. The experiment will then be submitted via the Web Service API to run on a large cluster of powerful servers, with only the modelling outputs being returned to the client for local visualisation.

Once the experiment is completed, the main application window provides a treeview of the experiment. The top level node of the tree is the experiment itself. Beneath this, mid level nodes represent each algorithm profile used for the experiment. The terminal nodes of the tree represent the individual models. Clicking on a model node provides a detailed report for the model. The model can also be visualised spatially using the map tab. The map tab uses an embedded GIS browser (based on Quantum GIS libraries) to allow the user to interactively pan and zoom around the geographic region of the output model.



Experiment Designer

Experiment Details

Name	Acacia cyclops CSM
Description	New Experiment created on sábado, 24 março, 2007
Coordinate System	Lat/Long WGS84

Occurrence Data

- [✓] Acacia_cyclops.txt
- [ ] Acacia_mearnsii.txt
- [ ] Acacia_saligna.txt
- [ ] Lantana_camara.txt

Algorithm(s)

- [ ] AquaMaps
- [ ] Bioclim
- [✓] Climate Space Model
- [ ] Distance to average
- [ ] Environmental Distance
- [ ] GARP (single run) – Des
- [ ] GARP (single run) – new

Model Settings

Creation Layers	Cru CL2
Projection Layers	Cru CL2
Output Format	/Users/tim/gisdata/CruCL2/stdevMeanTemp.asc

Figure 4: The experiment designer.

In the *Acacia cyclops* case study, Australia-only occurrence data was used to generate an fundamental niche model (Figure 4.) which was projected into a global climate dataset. In Figure 5, the results of this model are shown for south western Australia. The original occurrence points used to create the model are overlaid as black dots in the image. Areas in dark blue represent a poor match to the species' modelled fundamental niche. As the colours range through light blue, yellow and then to red, an increasingly good match to the species' modelled funamental niche is shown. By inference, if the species is planted in areas demarcated in red, then it probably has a good chance of survival. In Figure 6, the same model output is shown, but the area of interest is set to southern Africa. Based on the model, we can postulate that the Western Cape region of South Africa represents a similarly optimal habitat for *Acacia cyclops*. *Acacia cyclops* is indeed a problematic invasive plant species in the Western Cape, having been introduced there to stabilise sand dunes (11).

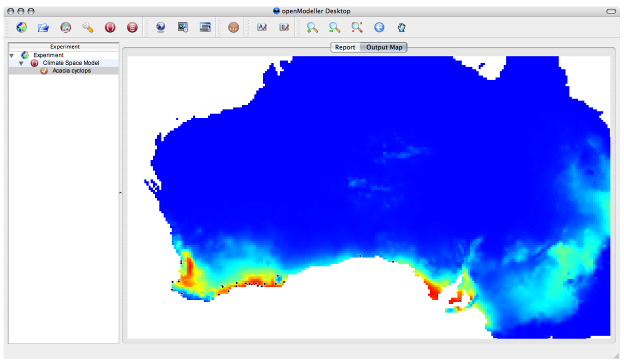


Figure 5: The model results for south western Australia.

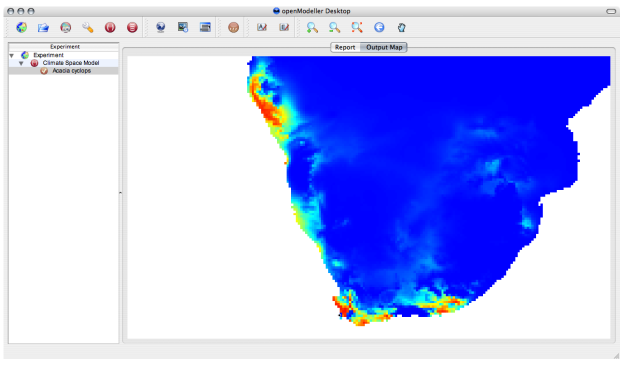


Figure 6: The model results for southern Africa.

Conclusion

openModeller provides solutions for users and developers interested in fundamental niche modelling. The *openModeller* library enables programmatic access to its capabilities. The console tools allow for scripting and interactive command line useage. The desktop application provides a user friendly graphical environment. The *openModeller* Web Services API provides a way to deploy *openModeller* on powerful servers (and in the near future, clusters). The plugin approach to implement different modelling algorithms allows the user to experiment with different modelling techniques without having to learn a new application each time, or having to prepare input data in a different way for each application. The algorithm plugin framework allows algorithm developers to focus exclusively on the core implementation of the algorithm without needing to 'reinvent the wheel' by writing data input / output routines and user interface logic.

The *openModeller Desktop* application provides an environment where all aspects of data preparation, experiment design, running an experiment and visualising the results of an experiment, can be carried out. *openModeller Desktop* can be compiled on all three major desktop operating systems (MS Windows, Mac OS X, GNU/Linux), and is freely available under the GPL.

The *openModeller* team is always eager to collaborate with other projects, users and developers. We encourage you to download and use the software and we welcome any feedback. We provide support via mailing lists and IRC - visit our home page for more details.

Bibliography

[1] Council of Heads of Australian Herbaria. 2007. **Australia's Virtual Herbarium via Centre for Plant Biodiversity Research** In http://www.cpbr.gov.au/cgi-bin/avh.cgi (Accessed 25 March, 2007)

[2] Brewer, P.W. 2003. **Modelling the global distribution patterns of Leguminosae species in past, present and future climates** PhD Thesis, University of Reading, UK.

[3] Elith, J., Graham C.H., Anderson R.P., Dudik M., Ferrier S., Guisan A., Hijmans R. J., Huettmann F., Leathwick J.R., Lehmann A., Li J., Lohmann L.G., Loiselle B.A., Manion G., Moritz C., Nakamura M., Nakazawa Y., Overton J.McC., Peterson A.T., Phillips S.J., Richardson K.S., Scachetti-Pereira R., Schapire R.E., Soberon J., Williams S., Wisz M.S., Zimmermann N. E. 2006. **Novel methods improve prediction of species' distributions from occurrence data.** In ECOGRAPHY 29(2):129ï£¡151

[4] Hutchinson, G.E. (1957). **Concluding Remarks.** In Cold Spring Harbor Symposia on Quantitative Biology. 22: 415-42.

[5] Nix, H.A. (1986). **A biogeogaphic analysis of Australian Elapid snakes.** In Longmore, R. (ed.)Atlas of Australian Elapid Snakes. Australian Flora and Fauna Series 8: 4-15.

[6] Peterson, A.T., Bauer, J.T., Mills, J.N. 2004. **Ecologic and Geographic Distribution of Filovirus Disease.** In Emerging Infectious Diseases 10(1):40-47

[7] Phillips, S. J., Anderson, R. P., and Schapire, R. E., (2006). **Maximum entropy modeling of species geographic distributions.** In Ecological Modelling, 190, 231-259.

[8] Robertson, M. P., Caithness, N., and Villet, M. H. 2001. **A PCAbased modelling technique for predicting environmental suitability for organisms from presence records.** In Diversity and Distributions vol.7: 15-27.

[9] Scachetti-Pereira, R. (2002). **Desktop GARP.** In http://lifemapper.org.br/desktopgarp (Accessed on January 10th 2007).

[10] Soberon, J., and Peterson, A. T. 2005. **Interpretation of Models of Fundamental Ecological Niches and Species' Distributional Areas.** In Biodiversity Informatics.

[11] Stirton, C.H. (ed). 1983. **Plant invaders - beautiful but dangerous.** Department of Nature and Environmental Conservation, Cape Town.

[12] Thomas, C.D., Cameron, A., Green, R.E., Bakkenes, M., Beaumont, L.J., Collingham, Y.C., Erasmus, B. F. N., de Siqueira, M.F., Grainger, A., Hannah, L., Hughes, L., Huntley, B., van Jaarsveld, A.S., and Midgley, G.F., and Miles, L., Ortega-Huerta, M.A., Townsend, A., Phillips, O.L., and Williams, S.E. 2004. **Extinction risk from climate change** In Nature:427

[13] Yesson, C., Culham, A. 2006a. **Phyloclimatic Modelling: Combining Phylogenetics and Bioclimatic Modelling.** In Systematic Biology: in press.

[14] Yesson, C., Culham, A. 2006b. **A Phyloclimatic Study of Cyclamen.** In BMC Evolutionary Biology. 6:72

Tim Sutton
CRIA, Campinas, Brazil
http://openmodeller.sf.net
tim AT linfiniti.com

Case Studies

MapGuide Open Source in the San Francisco Urban Forest Project

Open Source Mapping Helps City See the Forest for the Trees

by Alex Fordyce and Charlie Crocker

Abstract

The City and County of San Francisco, the non-profit Friends of the Urban Forest (FUF) and Autodesk have launched the San Francisco Urban Forest Mapping Project that uses MapGuide Open Source technology as its central platform for a dynamic online map of the tens of thousands of trees in San Francisco's public spaces.

FUF first planted the seeds for a tree-mapping initiative that would engage the public and foster community involvement in urban forestry. The project evolved to address the city's interest in strategic planting that maximizes environmental benefit and minimizes costs and labor associated with tree care, maintenance and replacement. The Urban Forest Mapping Project supports tree planting and management by the Bureau of Urban Forestry (BUF) and FUF and it's paving the way for use of geospatial information in other city initiatives.

Dynamic Map Integrates Spatial Data and More

BUF and FUF previously managed forest data in separate systems, both heavily reliant on paper-based maps, which made it difficult to share information. BUF employees in the field recorded tree location information on paper, and that data was then entered into a database along with information from permits issued for privately maintained trees. Updates would be collected from routine tree maintenance and entered, as well.

By BUF's best estimate, the city has about 100,000 trees planted in public rights-of-way. But the number of trees and permits stored in BUF's database amounts to just under half of that total, while FUF estimates it has recorded information for 41,000 trees that it has planted.

It was clear to both organizations that a more comprehensive and accurate inventory of the city's urban forest would help BUF and FUF better fulfill their shared goals. Hence, the Urban Forest Management system comprises an integrated database, a mapping tool and online reporting applications running in a hybrid open-source and proprietary software environment.

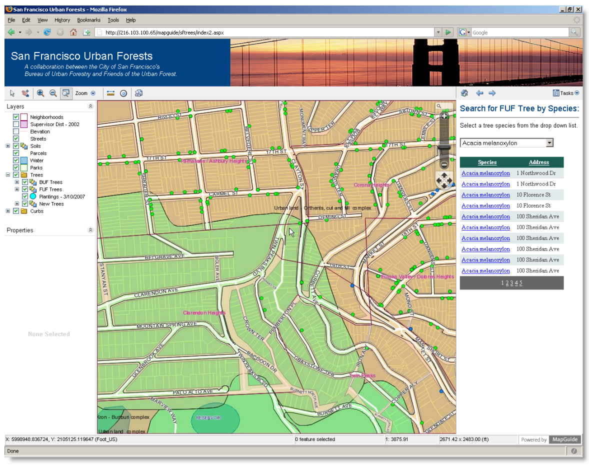
"Using a blended model of open source and proprietary technologies, we were able to create a system that met all our development and operations needs", said Greg Braswell, IT and GIS manager of the San Francisco Department of Public Works Bureau of Engineering. *"With MapGuide Open Source, we receive an enhanced level of collaboration tools and support for data sources and geocoding from the open source development community beyond what commercial (proprietary) vendors offer."* The new San Francisco Urban Forest Map accesses a centralized repository of urban forest inventory data through the Web. A shared database schema accommodates integrated tree attribute data from BUF and FUF, as well as real-time updates to spatial data made by BUF and FUF staff.

System Combines Open Source, Proprietary Technology to Meet City's Needs

Tree attribute information and tree point spatial data are both stored in a Microsoft SQL Server database. Attribute information is accessed using ASP.NET `C#`, while point locations are mapped directly from the database using the Open Source FDO data access technology provided in MapGuide Open Source. BUF and FUF staff already skilled on the Microsoft platform will not require additional training to maintain the system.

At the same time, BUF and FUF recognized that development of the solution in the .NET application environment would contribute to the geospatial industry's knowledge about application development and open source. The project is taking advantage of other open-source community efforts as well. Because the city uses MrSID for satellite image compression, the Urban Forest Map project team used Frank Warmerdam's MapGuide Open Source raster extension to support this proprietary raster format.

Urban forest spatial data and inventory details are combined with the city of San Francisco's base map data for streets, land parcels, soil types, and other map layers to create an interactive, Web-based map. The project also offered a good opportunity to take advantage of the application performance gained by using MapGuide Open Source's SDF format - a spatial file format streamlined for web-based environments. Land base map layers, which require less-frequent updating than the tree points, are stored as SDF files, while more dynamic tree information is geocoded and stored in a database Storing tree point information in a database allows other applications in the organization to access the tree data and make updates. Those updates are then seen in real-time on the Web application. Future enhancements to the system will allow authorized users to correct geocoding errors by moving tree points to more accurate locations. By overlaying a satellite image onto the map users can find a discrepancy between a geocoded tree and its actual location on a satellite image.


San Francisco Urban Forests - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://216.103.100.65/mapguide/sftrees/index2.aspx
San Francisco Urban Forests
A collaboration between the City of San Francisco's Bureau of Urban Forestry and Friends of the Urban Forest.
Zoom
Tasks
Layers
Neighborhoods
Supervisor Dist - 2002
Elevation
Streets
Soils
Parcels
Water
Parks
Trees
BUF Trees
FUF Trees
Plantings - 3/10/2007
New Trees
Curbs
Properties
None Selected
Search for FUF Tree by Species:
Select a tree species from the drop down list.
Acacia melanoxylon
1 2 3 4 5

Species	Address
Acacia melanoxylon	1 Northwood Dr
Acacia melanoxylon	1 Northwood Dr
Acacia melanoxylon	10 Florence St
Acacia melanoxylon	10 Florence St
Acacia melanoxylon	100 Sheridan Ave
Acacia melanoxylon	100 Sheridan Ave
Acacia melanoxylon	100 Sheridan Ave
Acacia melanoxylon	100 Sheridan Ave
Acacia melanoxylon	100 Sheridan Ave
Acacia melanoxylon	100 Sheridan Ave

Figure 1: The San Francisco Urban Forest Mapping System quickly maps BUF and FUF trees based on an address location search.

Business rules and logic were programmed using ASP.NET `C#` to define and control application activity according to user, user role and to track edit history. While all users share some functionality when working with the application, other authorized users are offered greater functionality. For example, users of the general public can browse the urban forest searching and viewing tree data according to various criteria (for example: address, neighborhood and species). The general public can also zoom, pan and toggle map views, and add trees with such relevant information as address, photos, comments and contact information. Authorized users are offered a greater set of functionality and may edit and update tree-related data.

Mapping Project Extends Life, Value of Data

The San Francisco Urban Forest Mapping System is expected to save the city money by providing the means to inventory and map past, present and future tree locations; calculate costs and benefits of the urban forest as a whole or in specific areas, identify effective strategies for tree planting and maintenance, and streamline processes such as permit applications. Future enhancements of the Urban Forest Mapping System hope to help facilitate city planning by allowing urban forest managers to model and calculate a full cost-benefit analysis and target a strategic approach for maximum advantage in San Francisco's many microclimates.

Likewise, FUF and the City leadership also aim to use the tool to engage San Francisco community members in urban forestry initiatives and greening efforts. The community can add trees to the map; this information is captured in a separate database for validation. When their information is confirmed, these trees are added to the Urban Forest database.

The use of open source mapping technology may reach beyond the San Francisco's urban forest canopy. The project team expects the program can also be repurposed by other cities, counties and public works organizations to map and manage any number of assets, beyond trees. Figure 1 shows trees surrounding city hall

For more information on the San Francisco Urban Forest Mapping Project, visit http://www.urbanforestmap.org.

Alex Fordyce
Founder and Senior GIS developer, Online Mapping Solutions LLC
afordyce AT ix.netcom.com

Charlie Crocker
Senior Product Manager, Geospatial Solutions, Autodesk
charlie.crocker AT autodesk.com

Integration Studies

Producing Press-Ready Maps with GRASS and GMT

by Dylan Beaudette

Overview

The production of high quality, printable maps is an important component of most geographic analysis. While GRASS has not traditionally been suited for the production of printed materials, commands like `ps.map` and more recently a helpful wrapper `G-ps.map`, make it possible to output high quality Postscript maps. However, the rather limited syntax and capabilities of the `ps.map` approach often result in a map that must be finalized in a DTP (desktop publishing) application such as Inkscape, Scribus, or The Gimp. The map composer functionality found in QGIS may be a good alternative for map production in the near future, but the current version is lacking in terms of flexibility, stability, and the ability to export data in formats such as EPS or PDF. With such limited options for the production of high quality printed materials when using an entirely open source workflow, numerous people have cobbled together the necessary glue required to interface GRASS to the various programs included with the Generic Mapping Tools package. The GMT project is best described by its authors as :

> GMT is an open source collection of 60 tools for manipulating geographic and Cartesian data sets (including filtering, trend fitting, gridding, projecting, etc.) and producing Encapsulated PostScript File (EPS) illustrations ranging from simple x-y plots via contour maps to artificially illuminated surfaces and 3-D perspective views. GMT supports 30 map projections and transformations and comes with support data such as coastlines, rivers, and political boundaries. GMT is developed and maintained by Paul Wessel and Walter H. F. Smith with help from a global set of volunteers, and is supported by the National Science Foundation. It is released under the GNU General Public License.

This article describes the combined efforts of several people (David Finlayson, Hamish Bowman, Brent Wood, myself, and others) to create a better means of interfacing GRASS to GMT. Several permutations of a `*.out.gmt` style command have been created, using both Python and Bash scripting, to perform the required export, configuration, and execution of GMT commands. After reading this article you should have a general idea of how to compose simple maps in GMT, along with some template scripts for using data exported from GRASS. A follow-up article will present Spearfish-specific (the standard sample GRASS dataset) examples of map creation with GMT.

GMT

The Generic Mapping Tools applications can be installed on just about any platform using either prepackaged binaries or by compiling from source code. The GMT tools can also be installed from your favorite distribution's package management system (e.g. aptitude), however this method is not recommended as the package is not likely to be current. If you are on a UNIX-like operating system compiling from source is a relatively simple process, and ensures that you have a current release. Assuming that you have a working development toolkit (gcc, make, etc.) navigate to the 'Download' page [51], and follow the directions. GMT is distributed with a comprehensive manual, map-making tutorials, and basemap data at several scales. Once you have a functional copy of GMT installed on your machine, and have downloaded the example data [52] you can follow along with the examples contained in this article. This archive also contains a couple scripts which elaborate on several of the examples included within this article. The script `template.sh` can be used as a starting point for an automated approach to exportng and plotting GRASS raster and vector data with GMT tools. Note that this script is a starting point and far from complete.

GMT commands are run from the shell (with an intricate set of flags, switches, and other arguments) and send the resulting Postscript fragments to standard output. The default behavior of all GMT applications can be adjusted with the `gmtset` command. This command is usually used prior to any commands which produce output, so that key elements such as paper size, font spacing, etc. are established. The first plotting command run (usually `psbasemap`) creates the initial output file with the standard ">" operator , while each subsequent call adds to this file with the double-">" append operator. The `-K` flag is used with all GMT commands prior to the last command, so that Postscript output file is not prematurely finalized. We will use some sample data collected from a Mapserver application to illustrate various aspects of GMT use, along with some ideas on how to couple GRASS and GMT.

Sample Application

```
# define some global settings:
gmtset ANNOT_FONT_PRIMARY Times-Roman \
HEADER_FONT_SIZE 16 \
ANNOT_FONT_SIZE_PRIMARY 12 \
LABEL_FONT_SIZE 14 \
BASEMAP_TYPE plain \
PLOT_DEGREE_FORMAT DF \
PAPER_MEDIA letter+

# sample map centered on the western USA
pscoast -JB-116/36/30/42/7i \
-R-125/-108/31/44 -B5 \
-Gwhite -W0.5p \
-A250 -Dh -Na -Xc -Yc -P -K > query_centers.eps

# plot mapxy points:
psxy sample_data/mapxy_locations.latlong -J -R \
-Sc0.075c -W1/1/200/1 -G1/200/1 -O -K >> query_centers.eps

# plot ka-map points
psxy sample_data/ka-map_locations.latlong -J -R \
-Sc0.075c -W1/255/1/1 -G255/1/1 -O >> query_centers.eps
```

The example above creates a plot of the western USA, with red and green dots symbolizing location-specific activity on a website. The map frame is created with the `psbasemap` command, coastline and political boundaries are created with `pscoast`, and points are created with the `psxy` command. Point locations are stored as simple, longitude-latitude pairs in text format, while the coastline is drawn from the GMT built-in data set. The output can be seen in Figure 1.
Full explanations of the various command line options can be found on the manual page of each command. In addition, a more complete example GMT session performed outside of GRASS can be found on the author's website [53]. In the above example, the projection and region parameters were manually defined by the user. Additional formatting elements (such as the `-P` flag for portrait layout) must be manually planned and defined on the command line. Thus the automatic generation of an output map depends, in part, on a dynamic approach to defining the options passed to GMT commands. A scripting language, capable of extracting GRASS region infor-

[51] http://gmt.soest.hawaii.edu/gmt/gmt_download.html
[52] http://169.237.35.250/~dylan/GRASS/newsletter/sample_data.tar.gz
[53] http://casoilresource.lawr.ucdavis.edu/drupal/node/102

mation from a running GRASS session, is an ideal approach to automating this task.

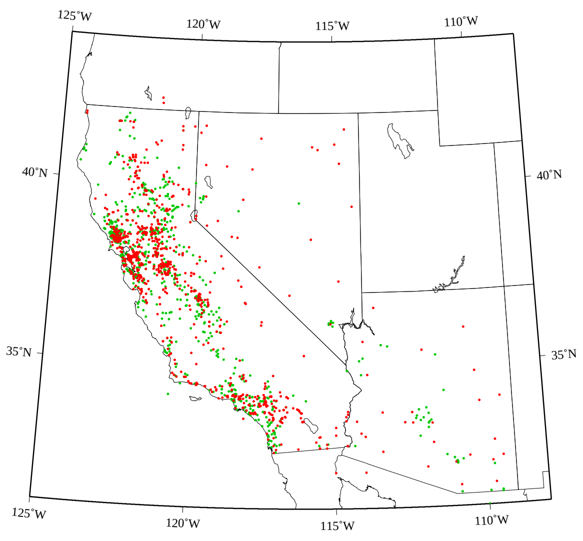


Figure 1: Query locations, symbolized by active mapping application. Green dots represent Landview-based Mapserver application, red dots represent Ka-map-based Mapserver application.

Coupling GRASS and GMT

The general approach to interfacing GRASS and GMT can be summarized as follows:

- collect information from the user on:
  - input data
  - formatting options
  - output options
- collect geographic parameters from the current GRASS region
- calculate key layout parameters
  - projection information
  - scale restrictions
  - tick intervals
- export GRASS raster data to GMT .grd format
- export GRASS vector data to GMT ASCII format
- run GMT commands with above settings

Collecting Information from the User

The GRASS script parser provides a convenient approach to the creation of customized GRASS commands, without the need to understand C programming. A carefully crafted script can be used to pass details about a GRASS dataset along with information on the current region settings to GMT commands, using the familiar syntax and GUI structure of a GRASS module. With these options saved to local variables it is possible to alter default settings such as paper size, font size, and text orientation by running the `gmtset` program with our local variables substituted by the script interpreter [54].

Collecting Geographic Information

Extracting key information from the current region settings (for later use in constructing parameters for GMT) can be accomplished in a number of ways. One such example involves `g.region` and unix text processing tools, thanks to Bruce Raup for the elegant `eval` suggestion. Note that all of the following code snippets require that you are in an active GRASS session.

```
# need to be in an active GRASS session for this to work.
#
# region extents: used with -R flag
# resolution: used with the -I flag
#
# elegant approach to extracting extent and
# region resolution as suggested by
# Bruce Raup - National Snow and Ice Data Center
#
region_assignments=`g.region -g`
eval $region_assignments

# setup flags for GMT commands
region="-R$w/$e/$s/$n"
inc="-I$ewres/$nsres"

# extract extent values for calculating map aspect ratio
extent_assignments=`g.region -ge`
eval $extent_assignments

# preserve aspect ratio from UTM E,N coordinates:
aspect_ratio=`echo $ns_extent $ew_extent
| awk '{printf("%f", $1 / $2)}'`

# calculate the map length based on the
# original aspect ratio
map_length=`echo $map_width $aspect_ratio
| awk '{printf("%f", $1 * $2)}'`
```

[54]More information on the GRASS script parser can be found at http://grass.itc.it/gdp/html_grass63/g.parser.html

```
# calculate the y location (in paper units)
# for the field sheet title
field_sheet_title_y=`echo $map_length
| awk '{printf("%f", $1 + 0.33)}'`

#compile the projection string, with
# width/length variables
#linear projection, width inches / length inches
projection_string="X${map_width}i/${map_length}i"
```

A more robust approach would involve the GRASS-Python bindings to directly extract region parameters. Once the region data has been extracted and saved to local variables, we can perform calculations required to derive other parameters such as map scale, map aspect ratio, map size on paper, etc. The region settings are used to create the *plotting region* flag (`-R$w/$e/$s/$n`), often used only in the first plotting function. All subsequent plotting functions can inherit these values by specifying blank `-J` and `-R` flags along with the `-O` (overlay) flag.

Map Projections in GMT

The traditional approach to using the GMT tools involves plotting data, encoded in longitude-latitude pairs, in a projection specified with the `-J` flag. Coordinates are transformed as the output Postscript is being generated. As noted in the first code snippet above, an Albers Equal Area projection centered at (116°W, 36°N), having standard parallels at 30°N and 42°N, and which is 7 inches wide is encoded as `-JB-116/36/30/42/7i`. See the GMT manual for a full listing supported projections and their parameters.

When working within GRASS, data is usually (but not always) in some cartesian projection and thus the special *linear* projection mode in GMT should be used. Once a suitable map width and length (in this case using inches) have been established the linear projection parameters are encoded as `-JX${map_width}i/${map_length}i`. Note that imperial or metric units can be used within GMT and switching between them only invloves adding a 'c' for centimenters or 'i' for inches after the actual number.

Exporting and Plotting GRASS Raster Data

Exporting GRASS raster data to a GMT-compatible format can be accomplished with a combination of `r.out.bin` and `xyz2grd` (included in the GMT distribution). Proper use of `xyz2grd` requires knowledge about input raster data: i.e. floating-point or integer, etc. For a CELL raster exported with `r.out.bin`, the flag `-ZTLh` would be used to instruct `xyz2grd` that the input data stream is scanline oriented short integer data. For FCELL or DCELL maps the corresponding flags would be `-ZTLf` or `-ZTLd` respectively. The `-F` flag instructs `xyz2grd` to use pixel-registration of grid cells, the same registration used for GRASS rasters. A simple case statement can be used within our example script to automatically create the call to `xyz2grd`. An optional flag in our example script can be used to toggle this step to facilitate tweaking other parameters without the time consuming process of exporting raster data. The author has found that setting the region to an integer resolution results in the most reliable operationof `xyz2grd`.

```
# get the map type of a given raster
MAP_TYPE=`r.info -t "$output_raster" | cut -f2 -d'='`

# export the raster based on its type,
# region settings from above example
# and resolution settings above example
case "$MAP_TYPE" in
CELL)
r.out.bin input=$output_raster output=- null=-9999 \
| xyz2grd -G$output_raster.grd \
$region $inc -ZTLh -F -N-9999 ;;
FCELL)
r.out.bin input=$output_raster output=- null=-9999 \
| xyz2grd -G$output_raster.grd \
$region $inc -ZTLf -F -N-9999 ;;
DCELL)
r.out.bin input=$output_raster output=- null=-9999 \
| xyz2grd -G$output_raster.grd \
$region $inc -ZTLd -F -N-9999 ;;
esac
```

Once raster data has been exported to .grd format it is nearly ready for use with GMT image plotting commands such as `psimage`. This command can either use a single grid file, along with a color palette (.cpt) file, or color-separate red, green, and blue channels [55].

Exporting GRASS Color Table Data

Currently there are no *simple* methods available for converting GRASS color table information into a suitable color palette file for GMT. The existing implementations are able to closely approximate the .cpt format, however the author has not been able to successfully use these automatically generated files. A working conversion script (preferable written in a

[55]The specifications for GMT color palette files can be found at http://www.soest.hawaii.edu/GMT/gmt/doc/html/GMT_Docs/node57.html

high level scripting language such as Python) will be posted to the GRASS wiki as soon as it is completed [56].

GMT provides several tools for manually creating a color palette file based on a user-supplied data range, or the data range as read from a GMT grid file. While this is not an ideal approach, it can be useful when one of the standard GMT color palettes is appropriate. In the special case where the exported image is grayscale, the standard "gray.cpt" color palette can be used to create a new cpt file with the makecpt program.

```
# make a suitable color palette for a grayscale
# CELL raster, the grid values are integers
# ranging from 0-255
makecpt -Cgray -T0/255/1 -V > doqq.cpt
```

When working with color channels that have already been separated (e.g. landsat or other multispectral imagery) it is possible to use psimage without a color palette file, instead specifying red, green, and blue input grid files. This approach is most convenient when working with data that has been collected in separate bands. For single band data, color separates can be created with r.mapcalc.

```
# get color-seperates ready for GMT

r.mapcalc "some_raster.red = r#some_raster"
r.mapcalc "some_raster.blue = b#some_raster"
r.mapcalc "some_raster.green = g#some_raster"
```

Exporting and Plotting GRASS Vector Data

The GMT vector format is based on a very simple, vertex-based ASCII format, designed primarily to store geometry data. However, with some scripting it is possible to encode simple attribute data within this format for the production of thematic maps. An alternative approach involves exporting vector data that has been pre-filtered with a GRASS tool such as v.extract, into several files where each represents a specific class. Using the first approach (encoding attributes within the GMT vector file) the symbology for each point, line, or polygon is encoded within the file, whereas the symbology is set with flags on the command line for the second approach.

Point Data

Exporting point data is the simplest GRASS → GMT vector conversion. In most cases, where a set of points is to be plotted in GMT using a fixed symbology, the output of v.out.ascii or v.out.ascii.db can be directly plotted with the GMT command psxy. Note that filtering the output from either command with UNIX text processing tools is a convenient method for converting attributes into symbology or adjusting labelling.

```
# export from GRASS to ASCII format
v.out.ascii in=points fs=" " > points.xy

# plot points with a white outline and blue fill,
# using a circular symbol 0.125 inches in diameter
# appending the resulting Postscript to 'outfile.eps'
psxy points.xy -R -J -M -Sc0.125 -G255/1/1 \
-W1/255/255/255 -O -K >> output.eps
```

Labeling point data can be accomplished by first filtering the output from v.out.ascii with a simple awk script, then using the output with the pstext command. The labeling format used by GMT is documented on the pstext manual page, however a simple explanation is given within the sample code below.

```
# export from GRASS to ASCII format with selected
# attributes ('ID' column)
# filter output with awk, re-ordering columns,
# and inserting label properties into the form:
# x_coord, y_coord, font_size, rotation_angle,
# font_number, label_offset, label_text
v.out.ascii.db in=points columns=ID | awk -F"|"
'{print $2, $3, 10, 0, 4, "BL", $4}' \
> points_with_labels.xy

# plot label text from geometry and label
# properties stored in 'points_with_labels.xy'
# label text will be blue and offset by 0.1 cm
# in the horizontal and vertical direction
# specified in column 5 of the input file
pstext points_with_labels.xy -R -J -Dj0.1c/0.1c
-G0/0/255 -K -O >> outfile.eps
```

Line and Polygon Data

There is currently no direct method for converting GRASS line and polygon data into a GMT compatible format. However, through the use of an external application called shp2gmt [57], it is possible to create GMT compatible vector files by first exporting GRASS data to shapefile and then converting with shp2gmt. The text output from shp2gmt contains all

[56]Further testing and adaption of David Finlayson's r.out.gmt.py script may be the quickest approach.

[57]This small utility program originally written by (Frank Warmerdam), was modified by (Mark Fenbers) to include attribute information from the shapefile as well. The code for the modified shp2gmt can be found at http://www.arcknowledge.com/gmane.comp.gis.gmt.user/2004-01/msg00121.html

of the attribute data stored in the input shapefile along with vertex coordinates, and can be easily filtered with awk to include symbology parameters. An example GRASS session illustrating this type of operation is listed below.

```
# export from GRASS to shapefile format
v.out.ogr -e in=lines dsn=. olayer=lines
v.out.ogr -e in=polys dsn=. olayer=polys

# convert from shapefile format to GMT format
shp2gmt lines.shp > lines.xy
shp2gmt polys.shp > polys.xy

# optionally filter GMT vector file to include symbology
awk '
{
# all multi-record line segments with the
# attribute 'some_value'
# will be plotted with a red pen
if ($0 ~ /some_value/) printf "> -Wred\n"
# ... add more lookups for the other classes
# for lines that do not match, just print them
# verbatim (i.e. the vertex data)
else print
}
' lines.xy > lines-thematic.xy

# plot lines, with symbology set within the GMT vector file
psxy lines-thematic.xy -R -J -O -K >> outfile.eps
```

Labeling line or polygon features follows the same general approach as labelling point features, through the use of pstext. For simple maps, an almost automatic approach to labelling polygon data involves exporting polygon centroids with v.out.ascii.db, as listed above. Labelling line data requires more user interaction, as each label needs a coordinate and angle associated with it. With a little work it is possible to use the file created by v.label to produce a GMT-compatible set of label placement instructions. An example awk script, used within a GRASS session, is presented below.

```
# create the GRASS label file:
# rotating the labels to match the line segments
v.label -a map=trails column=trail_name labels=trails.lab

# convert the GRASS labels format to GMT format
awk '
BEGIN{FS="\n" ; RS="\n\n"}
{
split($1, e, ": ")
split($2, n, ": ")
split($15, r, ": ")
split($16, l, ": ")
# if there is no rotation, we need to add a 0-rotation
if(NF == 16) {rotation = r[2] ; label = l[2]}
else {rotation = 0 ; label = r[2]}
# create the GMT labeling instructions
print e[2], n[2], 10, rotation, 4, label
}
' DATABASE/LOCATION/MAPSET/paint/labels/trails.lab
> labels.xy

# plot with pstext
pstext labels.xy -R -J -Dj0.1c/0.1c -G0/0/255 -K -O
>> outfile.eps
```

While the above approach may work for simple maps, user intervention is usually required for more complex maps. A more robust approach for labeling point, line, and polygon features with advanced capabilities such as label collision detection is needed for a fully automated approach.

Miscellaneous Map Elements

Additional elements such as a context map (psbasemap with psxy), scalebar (psbasemap), or legend (pslegend) can be added after primary map components have been added to the output file. It is always a good idea to check that the last GMT command run does not use the -K flag, to insure that the output file is finalized. Through careful use of the -O, -X, and -Y flags it is possible to re-define the plotting region for each call to a GMT program. This approach works well for creating a mini-sized context map, within another finished map. Extending the first example, below is a more complete script illustrating the use of psbasemap to produce map elements such as a scale bar and north arrow, along with the creation of a mini-context map within a map.

```
# define some global settings:
gmtset ANNOT_FONT_PRIMARY Times-Roman \
HEADER_FONT_SIZE 16 \
ANNOT_FONT_SIZE_PRIMARY 12 \
LABEL_FONT_SIZE 14 \
BASEMAP_TYPE plain \
PLOT_DEGREE_FORMAT DF \
PAPER_MEDIA letter+

# sample map centered on the western USA
pscoast -JB-116/36/30/42/7i \
-R-125/-108/31/44 -B5 \
-Gwhite -W0.5p \
-A250 -Dh -Na -Xc -Yc -P -K
> query_centers_2.eps

# plot mapxy points:
psxy sample_data/mapxy_locations.latlong -J -R \
-Sc0.075c -W1/1/200/1 -G1/200/1 -O -K
>> query_centers_2.eps

# plot ka-map points
psxy sample_data/ka-map_locations.latlong -J -R \
-Sc0.075c -W1/255/1/1 -G255/1/1 -O  -K
>> query_centers_2.eps

# add scalebar
```

```
psbasemap -J -R -Lf-119/32.5/32/150k:"Kilometers": \
-O -K -P -V >> query_centers_2.eps

#make a context map
pscoast -JB-116/36/30/42/7i \
-R-130/-100/25/50 \
-X0.25i -Y1.75i \
-Gwhite -W0.5p \
-A250 -Di -Na -P -K >> query_centers_2.eps

# add the region box
psxy context_box.xy -M -JB -R  -W3/1/1/0 -P -O
>> query_centers_2.eps
```

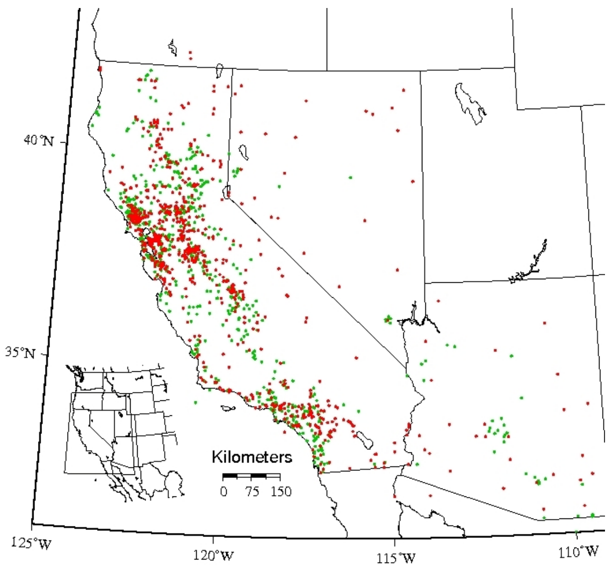


Figure 2: Addition of a context map and scalebar.

Current Limitations

The current unix shell approach to interfacing GRASS and GMT has several serious disadvantages: lack of portability to non-unix systems, lack of sophisticated mathematical operators, and a dependancy on several helper tools such as `awk`, `sed`, `grep`, etc. These disadvantages are particularly annoying when trying to automatically construct map elements such as tick intervals or scale-based decorations. The examples in this article use manually-defined tick intervals, however when creating maps from GRASS data (which is usually in a projected coordinate system) it is time consuming to manually set the annotated and non-annotated tick intervals for each map. See the `template.sh` script included in the sample data archive for ideas. Again, tighter integration with GRASS via the Python bindings would solve many of the above problems. Until a fully-Python implementation is complete, estimation of automatic tic intervals can be simplified with the following perl script.

```
#!/usr/bin/perl -w

# the first argument is the maximum extent in map units
# divide by a resonable number of annotations per edge
$x = $ARGV[0] / 10;

print round_up($x) , "\n";

# round to a resonable scale
# found at:
# http://www.perlmonks.org/?node_id=599865

sub round_up {
  my $n = shift;
  my $scale = 10**int(log($n)/log(10));
  $n = 9 if $scale == 1; #magic for single digits
  if ($n > $scale) {
    $n = int($n/$scale+1)*$scale;
  }
  $n;
}
```

Conclusions

This article summarizes the current condition of converting GRASS data into GMT format for the production of high quality, Postscript maps. Although several templates and examples exist to automate this process, for complex map production considerable modifications by the user are required. Given sufficient progress on projects like the GRASS GUI and Python bindings, it may be possible to create a more complete system for streamlining the GRASS → GMT workflow. There is currently an effort lead by Brent Wood to overhaul the GMT vector format which would simplify the process of thematic mapping with GMT. In addition Brent is working on including write-support for the GMT vector format into OGR (GDAL). Once completed, vector export from GRASS to GMT could be as simple as `v.out.ogr format=GMT`. Proposed work by Wolf Bergenheim, on a general label collision detection and correction algorithm for `v.label`, would be an excellent solution to complex labeling tasks in both GRASS and GMT. Tune in next time for an in-depth example of creating a complex map using GRASS data from the Spearfish sample data.

Dylan Beaudette
University of California at Davis
http://casoilresource.lawr.ucdavis.edu
debeaudette AT ucdavis.edu

Using the R— GRASS interface

Current status

by Roger Bivand

Introduction

Interfaces between GRASS and R, the open source data analysis and statistical programming environment, have existed for some time. Details of the interface between GRASS 6 and R were described two years ago in Bivand (1), but since then things have got a lot simpler.

Intermediate temporary files are the chosen solution for the GRASS 6 interface: **spgrass6**, using shapefiles for vector data and BIL binaries for raster data. R is started from within a GRASS session from the command line, and the **spgrass6** loaded with its dependencies, with the R interface being used to access and update GRASS data.

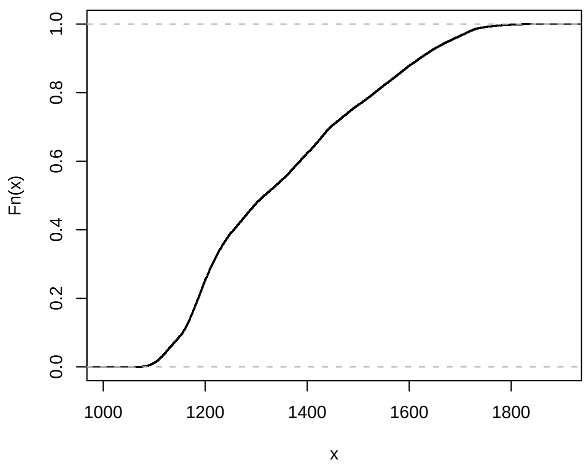


Figure 1: Empirical cumulative distribution function of elevation for the Spearfish location.

Installing the interface package

The GRASS 6 interface is available from CRAN, the Comprehensive R Archive Network. It depends on three packages, andm if not already available, these (**sp**, **maptools** and **rgdal**) should be installed within R using the dependencies= argument:

```
> install.packages("spgrass6", dependencies = TRUE)
```

To install on a server not running a graphical interface, set the CRAN mirror first with:

```
> chooseCRANmirror(graphics = FALSE)
```

The only potential difficulties for installation of these packages from source on Linux, Unix, or Mac OS X are with **rgdal**, because of its external dependencies on GDAL and PROJ.4 libraries. On Unix/Linux, note that development files for GDAL are required, not just GDAL itself, if your GDAL was installed binary rather than from source. All the other packages are available as binaries for Mac OS X users, but **rgdal** is not. Notes for Mac OS X users about installing **rgdal** are to be found on the Rgeo website — see under **rgdal**. Windows binaries are available for all the packages, and work with GRASS 6 under Cygwin.

Using the package

```
> library(spgrass6)
> gmeta6()
```

The examples used here are taken from the "Spearfish" sample data location (South Dakota, USA, 103.86W, 44.49N), perhaps the most typical for GRASS demonstrations. The gmeta6 function is simply a way of summarising the current settings of the GRASS location and region within which we are working. At the present stage of the interface, raster data transfer is done layer by layer, and uses temporary binary files. The readRAST6 command here reads elevation values into a SpatialGridDataFrame object, treating the values returned as floating point, and the geology categorical layer into a factor:

```
> spear <- readRAST6(c("elevation.dem",
+     "geology"), cat = c(FALSE, TRUE))
```

```
> summary(spear)
```

```
Object of class SpatialGridDataFrame
Coordinates:
              min     max
coords.x1  589980  609000
coords.x2 4913700 4928010
Is projected: TRUE
proj4string :
[+proj=utm +zone=13 +a=6378206.4
+rf=294.9786982 +no_defs
+nadgrids=/home/rsb/topics/grass63/grass-6.3.cvs
/etc/nad/conus
+to_meter=1.0]
Number of points: 2
Grid attributes:
  cellcentre.offset cellsize cells.dim
1            589995       30       634
```

```
2          4913715       30       477
Data attributes:
 elevation.dem          geology
 Min.   : 1066   sandstone:74959
 1st Qu.: 1200   limestone:61355
 Median : 1316   shale    :46423
 Mean   : 1354   sand     :36561
 3rd Qu.: 1488   igneous  :36534
 Max.   : 1840   (Other)  :37636
 NA's   :10101   NA's     : 8950
```

When the cat= argument is set to TRUE, the GRASS category labels are imported and used as factor levels; checking back, we can see that they agree:

```
> table(spear$geology)

metamorphic  transition     igneous
      11693         142       36534
  sandstone   limestone       shale
      74959       61355       46423
sandy shale    claysand        sand
      11266       14535       36561

> system("r.stats --q -cl geology",
+     intern = TRUE)

 [1] "1 metamorphic 11693"
 [2] "2 transition 142"
 [3] "3 igneous 36534"
 [4] "4 sandstone 74959"
 [5] "5 limestone 61355"
 [6] "6 shale 46423"
 [7] "7 sandy shale 11266"
 [8] "8 claysand 14535"
 [9] "9 sand 36561"
[10] "* no data 8950"
```

Figure 1 shows an empirical cumulative distribution plot of the elevation values, giving readings of the proportion of the study area under chosen elevations. In turn Figure 2 shows a simple boxplot of elevation by geology category, with widths proportional to the share of the geology category in the total area. We have used the readRAST6 function to read from GRASS rasters into R; the writeRAST6 function allows a single named column of a SpatialGridDataFrame object to be exported to GRASS.

The **spgrass6** package also provides functions to move vector features and associated attribute data to R and back again. The readVECT6 function is used for importing vector data into R, and writeVECT6 for exporting to GRASS:

```
> bugsDF <- readVECT6("bugsites")

> vInfo("streams")

    points      lines boundaries   centroids
         0        104         12           4
     areas    islands      faces     kernels
         4          4          0           0

> streams <- readVECT6("streams", type = "line,boundary",
+     remove.duplicates = FALSE)
```

The remove.duplicates= argument is set to TRUE when there are only for example lines or areas, and the number present is greater than the data count (the number of rows in the attribute data table). The type= argument is used to override type detection when multiple types are non-zero, as here, where we choose lines and boundaries, but the function guesses areas, returning just filled water bodies.

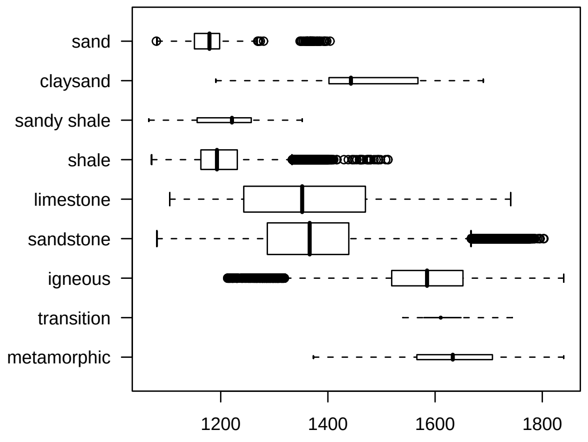


Figure 2: Boxplots of elevation by geology category, Spearfish location.

Because the mechanism used for passing information concerning the GRASS location coordinate reference system differs slightly between raster and vector, the PROJ.4 strings often differ slightly, even though the actual CRS is the same. We can see that the representation for the point locations of beetle sites does differ here; the vector representation is more in accord with standard PROJ.4 notation than that for the raster layers, even though they are the same. In the summary of the spear object above, the ellipsoid was represented by +a= and +rf= tags instead of the +ellps= tag using the clrk66 value:

```
> summary(bugsDF)

Object of class SpatialPointsDataFrame
Coordinates:
              min     max
coords.x1  590232  608471
coords.x2 4914096 4920512
Is projected: TRUE
proj4string :
[+proj=utm +zone=13 +ellps=clrk66
+datum=NAD27 +units=m +no_defs
+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat]
Number of points: 90
Data attributes:
      cat                   str1
 Min.   : 1.00   Beetle site:90
 1st Qu.:23.25
 Median :45.50
 Mean   :45.50
 3rd Qu.:67.75
 Max.   :90.00
```

This necessitates manual assignment from one representation to the other on occasion, and is due to GRASS using non-standard but equivalent extensions to PROJ.4.

There are number of helper functions in the **spgrass6** package, one `gmeta2grd` to generate a GridTopology object from the current GRASS region settings. This is typically used for interpolation from point data to a raster grid, and may be masked by coercion from a SpatialGrid to a SpatialPixels object having set cells outside the study area to NA. A second utility function for vector data uses the fact that GRASS 6 uses a topological vector data model. The `vect2neigh` function returns a data frame with the left and right neighbours of arcs on polygon boundaries, together with the length of the arcs. This can be used to modify the weighting of polygon contiguities based on the length of shared boundaries. Like GRASS, GDAL/OGR, PROJ.4, and other OSGeo projects, the functions offered by **spgrass6** are changing, and current help pages should be consulted to check correct usage.

Bibliography

[1] Bivand, R. S., (2005) Interfacing GRASS 6 and R. *GRASS Newsletter*, 3, 11–16, http://grass.itc.it/newsletter/.

Roger Bivand
Economic Geography Section, Department of Economics, Norwegian School of Economics and Business Administration, Bergen, Norway
http://www.r-project.org/Rgeo
Roger.Bivand AT nhh.no

Geospatial Processing via Internet on Remote Servers – PyWPS

PyWPS and Embrio

Jáchym Čepický and Lorenzo Becchi

Document OGC 05-007r4[58] *describes a process for offering geospatial operations over networks using Web Services. This paper introduces an example implementation – PyWPS. Even if the original target of PyWPS was to make modules of GRASS GIS accessible from Internet applications, in general it is possible to use any command-line oriented tool or any tool which has bindings to Python Programming language. With the help of PyWPS we can perform time consuming calculations on the server side, as well as build your real WebGIS application, running in a web browser. Let us describe how it works and if PyWPS could fit your needs.*

OGC Web Processing Service

Since the OGC Web Processing Service (WPS) standard is still relatively new and not as well known as, for example, its cousin the Web Map Service (WMS), we start by providing a brief overview of the standard here. The basic unit of the WPS is the *process* – a geospatial operation, with inputs and outputs of a defined type. The client communicated with the server with the help of three types of requests. The request can be sent to the server via HTTP GET with parameters provided as Key-Value Pairs (KVP) or via HTTP POST, with parameters supplied in a XML file. There are three types of requests that can be sent to the server:

GetCapabilities – Server responds with XML, describing server provider, fees, general description and giving a list of processes, prepared to be performed.

DescribeProcess – Server responds with XML, which describes concrete inputs and outputs type, so the client is able to formulate the *Execute* request.

Execute – Client requests the execution of a geospatial operation, with all required input data – the server process runs and informs the client (the user) of its progress.

The following are some illustrative examples of these requests. Let us assume we would like to perform line-of-sight calculation from defined x and y coordinates on a raster file, which can be obtained from a remote server. The process name will be *visibility*.

Basic usage of OGC Web Processing service

First we need to find out which calculations the server offers (see this link [59]) From the resulting XML it is clear that the process *visibility* is available on the server and the abstract tells us that it does what we would like. In the second step we need to find out what kinds of input and output the process requires or will send back (see this link[60])

- *x* of type *LiteralValue*
- *y* of type *LiteralValue*
- *maxdist* – maximum distance from the observer. Type *LiteralValue*, minimal allowed value is 0, maximal 5000 meters.
- *observer* – observer height. Type *LiteralValue*, minimal allowed value is 0, maximal 50 meters.
- *dem* – type *ComplexValue* – raster map of digital elevation model, on which the visibility should be calculated.

Now we can formulate the input request, send it to the server and look forward to the calculation results[61] The server will download the input digital elevation model, perform the line-of-sight calculation and return the resulting raster image back to the client.

[58]OGC Web Processing Service (WPS): http://www.opengeospatial.org/standards/requests/28

[59]http://pywps.ominiverdi.org/cgi-bin/wps.py?service=WPS&request=GetCapabilities

[60]http://pywps.ominiverdi.org/cgi-bin/wps.py?service=WPS&request=DescribeProcess&version=0.4.0&identifier=visibility

[61]http://pywps.ominiverdi.org/cgi-bin/wps.py?service=WPS&version=0.4.0&request=Execute&identifier=visibility&datainputs=x,602829.1875,y,4925326.875,maxdist=2000,observer=1.2,dem,http://somewhere/?some&service

Introduction to PyWPS

PyWPS is a relatively new project, started in April 2006. The original project goal was to make the connection between UMN MapServer and GRASS GIS as easily possible so that we could build a real WebGIS application able to perform, for example, interpolation of raster data or various digital elevation model analysis. Time has shown that even though GRASS GIS is a powerful tool, it is not necessarily the best or the only choice for all possible tasks. The design of PyWPS has changed so that it could be used without GRASS GIS in the background, but with any other tool or just with Python itself.

PyWPS is an implementation of OGS's Web Processing Service standard as defined in the document OGC 05-007r4. Currently, the complete standard is not yet supported, but around 95% of the standard is implemented and usable.

The project is built on a simple CGI script hoping to make the life of WebGIS coders as easy as possible. It provides functions to:

- Parse all input and create all output
- Perform basic validation of the input, like insuring type of LiteralValue input or maximum file size for the ComplexValue input, etc.
- Create and remove on-the-fly generated temporary files and directories, like temporary GRASS locations and mapsets and other files created as part of the calculation.
- Run other useful operations

The coder has to do only one thing: define the processes input and output, which is basically a script in the Python programming language. The process is one class *Process*, with one mandatory method *execute*, in which the calculation is provided. Input and output data are defined in a similar way, in a complex dictionary structure[62]:

```
{
    'Identifier':'maxdist',
    'Title': 'Maximal distance',
    'Abstract':'Maximal distance of visibility',
    'LiteralData': {
        'values':[ [0.,5000] ],
    },
    'dataType': type(0.0),
},
{
    'Identifier': 'dem',
    'Title': 'Digital elevation mode'
    'Abstract': 'Raster map with elevation model',
    'ComplexValueReference': {
        'Formats':["image/tiff"],
    }
},
```

The *execute()* method can then use e.g. GRASS modules directly:

```
def execute(self):
    # importing dem
    os.system("r.in.gdal in=%s out=dem" %\
                (self.datainputs['dem']))
    # setting region according to dem file
    os.system("g.region rast=dem")
    # lines-of-sight module
    os.system("r.los input=dem output=output \
            coordinate=%s,%s max_dist=%f \
            obs_elev=%d" % \
            (self.datainputs['x'],
            self.datainputs['y'],
            self.datainputs['maxdist'],
            self.datainputs['observer']))
    # exporting raster map
    os.system("r.out.gdal in=output out=out.tif")
    # setting the output value
    self.dataoutputs['output'] = "out.tif"
    return
```

As the source code is the best documentation, around ten example processes are distributed together with the PyWPS source, so the user can get a general picture of the process definition. There is also on-line and offline documentation available, which tries to describe the installation process and setup of your own processes.

Recently, a new class has been defined, which provides easy definition of process inputs and outputs as well as better support for GRASS GIS modules. Any web interface will be able to track progress of data imports derived directly from the `r.in.gdal` module. This improvement will be available in the next release of PyWPS.

Further development

The first "stable" version of PyWPS with the version number 1.0.0 was released in November 2006. Currently, the PyWPS development team is planning to release version 2.0.0 soon, with added functionality and a few bug fixes. Common effort in the development goes in three directions:

- Implementation of the complete OGC WPS standard
- Making the application as secure as possible, to avoid server compromise

[62]Note that this has been replaced by new methods *Add*Input()* in current *svn* version

- Making life of the process-coder as easy as possible

The PyWPS development team would also like to start discussion in the geospatial communities about defining process metadata. The OGC WPS standard defines the input types from the process point of view, however it does not say anything about input (or output) type from the user (interface) point of view. For example, coordinate x is of type *LiteralValue* but this does not describe x as a coordinate. So it could be useful to setup this input value with a mouse click in the map window rather then with the keyboard in some input form field. If you want to have a closer look at PyWPS, feel free to visit the PyWPS project page[63] where the source code as well as links to projects already using PyWPS are available.

Using ka-Map & PyWPS to create a GRASS WEB GIS

Ominiverdi[64] became involved with developing PyWPS after its initial release. Our first goal was to create a FOSS Web Client to access GRASS functions. PyWPS was not the first attempt to do a connection between GRASS and the Web but none of the others were based on open standards, making it impossible to create a shared platform.From the beginning we started planning two different projects: **Embrio** and **Wuiw**.

Embrio

This is the first implementation we had in mind: use of PyWPS to let UMN MapServer and its MapScript API to interact with GRASS.

Another important goal was a Web rich client and that's why we decided to base our efforts on ka-Map. Ka-Map uses MapScript to access the powerful set of features in UMN MapServer and offers a Google Maps-like navigation experience. The output of PyWPS, once the Process returns a map, can be in GeoTiff, for raster output, and in GML for vector output. Both formats are natively accessible by UMN MapServer and can easily be coupled with an existing map environment. In this way a request for the *Visibility* module can return a GeoTiff that is kept by ka-Map and inserted in the actual map coordinates system. Then a style, that can be an SLD, is applied to raster values and a temporary cache is created on the fly while tiles are sent back to the client. The temporary cache is related to the *sessionId*.

This system allows different modules to run in parallel and to compare their outputs using the ka-Map interface. Layer opacity control and layer vertical position can be useful to understand the resulting output. Even the query function is synchronized if the output is queryable. The *Visibility* module can output the incident angle with the point of view, the query function can return the value for each pixel clicked.

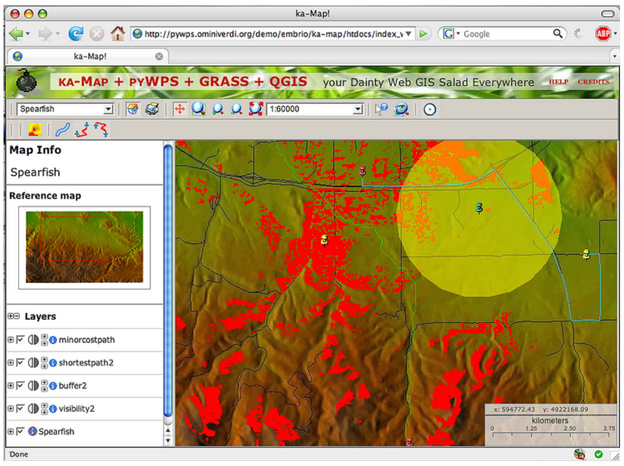


Figure 1: Screenshot of the most recent Embrio interface

Wuiw

Wuiw is still more of a concept than an application. It's intended to offer a Javascript API that connects the WPS to data resource as OWS and render the output independently from any mapserver application.

We are still evaluating current limits of the WPS draft that are not yet relating the input definition with a complex type definition. The first idea has been to use Metadata information to create this kind of interaction but there is the risk of developing a parallel dialect that will loose the benefits of standard interoperability.

We hope that the WPS path to version 1.0 will create a comfortable solution to this limitation.

[63]PyWPS project page: http://pywps.wald.intevation.org
[64]Ominiverdi website: http://www.ominiverdi.org

Further development

It is obvious that WPS, PyWPS, Embrio and Wuiw all have short histories. Even if most things are still to be done, the beginning is promising. Regarding Embrio's future, we plan to improve interaction with the WPS script process feedback and show a process progress bar. Many more GRASS modules can be developed and we hope to receive help from the GRASS community to achieve this.

Another important goal is to add more interaction with CLI (command line interface) accessible applications, including R, GDAL/OGR, etc. for geo statistics, format conversion and many, many other functions.

We also hope to add an AJAX CLI to interact with a protected system interacting through WPS.

Licenses

It important to note that this project uses many different software packages with at least two different licenses.

- GNU/GPL: GRASS, PyWPS, R
- MIT/BSD: UMN MapServer, MapScript, ka-Map, Embrio

Resources

- OGC Web Processing Service [65]
- PyWPS home page: [66]
- Last Embrio live example: [67]
- Embrio home page [68]

Jáchym Čepický
http://les-ejk.cz
jachym AT les-ejk cz

Lorenzo Becchi
http://ominiverdi.org
lorenzo AT ominiverdi com

Tikiwiki a GeoCMS

by Franck Martin

Overview

You may have heard of Content Management Systems (CMS), these are web based applications that allow you to share content over the web. The concept evolved by mixing wikis with forum based web sites. A wiki[69] web site allows fast creation of web pages using a simple formatting language. Links to new pages are first created, then the pages themselves. From this, forums, image galleries and blogs have been added. Tikiwiki[70] was created as an integrated CMS from the start as a more robust approach, allowing the interaction of all the features with each other. In Tikiwiki, you have a wiki, image galleries, file galleries, blogs, forums, articles, trackers and many more, all linked via a user and group management system to define permissions on objects. Tikiwiki is database independent, using *PHP ADODB*[71] and it uses *Smarty*[72], a template engine for PHP. It is then easy to personalise the look and feel of Tikiwiki so that your web site does not look like a default Tikiwiki site.

What is exactly Tikiwiki?

Tikiwiki is one of the first developed CMS, it is highly configurable but suffers from a large code base and a lack of modules. However, this disadvantage is also an advantage if you want to deploy all the functionalities in a CMS because they are tightly integrated rather than independent modules. Tikiwiki can also be extended via *mods*, a sort of plug-in

[65] http://www.opengeospatial.org/standards/requests/28
[66] PyWPS Home page: http://pywps.wald.intevation.org
[67] Live example: http://pywps.ominiverdi.org/demo/embrio/ka-map/htdocs/index_wps_qgis.html
[68] http://pywps.ominiverdi.org/
[69] Definition of wiki: http://en.wikipedia.org/wiki/Wiki
[70] Tikiwiki web site: http://tikiwiki.org
[71] PHP ADODB project: http://adodb.sourceforge.net/
[72] Smarty template engine: http://smarty.php.net/

system that allows you to extend functions. I chose Tikiwiki as a CMS because it was ranked as project of the month on Sourceforge and also because the developer community is very open. It was quite easy to add code to the code base. This is important when you start to contribute to a project.

Tikiwiki has an extensive security system and current development is focusing on implementing *AJAX* in the code to make it more efficient. The other strong point of Tikiwiki is internationalisation. There are several languages for the interface and it is easy to add your own language via the php equivalent of *gettext*. Also Tikiwiki supports the UTF8 character set and right to left writing. You can mix several languages in the same page as well as provide integrated translation for wiki pages. Finally, Tikiwiki is highly customisable via its *CSS* and *Smarty* templates.

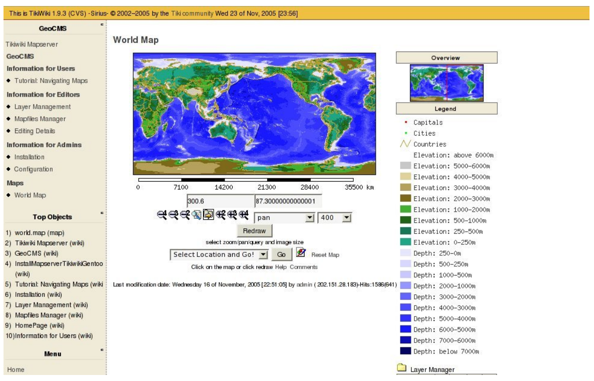

This is TikiWiki 1.9.3 (CVS) -Sirius- © 2002–2005 by the Tiki community Wed 23 of Nov, 2005 [23:56]
GeoCMS
World Map
Overview
Legend
Capitals
Cities
Countries
Top Objects
Menu
Layer Manager

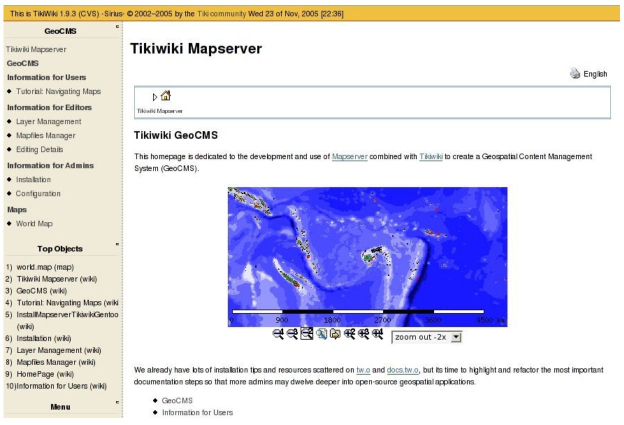
Figure 1: A standalone map in Tikiwiki.

A GeoCMS, What is it?

Since the success of Google Maps, maps are becoming an important feature of any web site. The success of Google Maps is due to the publication of its API, allowing any user to complement a Google map with point data, like the location of a user trip. Maps have been available for a long time on the web, like *Mapquest*, but were not available for customisation to the public. A ***G**eospatial **C**ontent **M**anagement **S**ystem* (GeoCMS) is a CMS with geographical objects and map rendering capabilities. Despite having on your web site an online interactive web map, that you can zoom in, resize and query objects, you have also geographical objects that you can represent on the map. These maps can be standalone (see fig. 1) or included in a wiki page (see fig. 2). For instance, you can request your registered user to input their location and have them represented as a dot on the map. You can create an image gallery where each image has a location and have them represented as a dot on the map, to show your last trip, important monuments, geological features, and so on.

With any geographical layer there is the problem of knowing enough information about the layer (metadata): who created the layer, when, what is the quality of the data, in what projection system the objects are represented, etc.. It is important to have this information to build a level of trust in the data. If you have a population layer and a disaster strikes on a specific region, you must have relative trust in the data when preparing your response. Linking layers on the map with wiki pages is possible inside Tikiwiki. The metadata is therefore simple to enter and the specialist can then describe interesting features represented on the map.

Tikiwiki also provides an entire module to upload and process geographical data layers to put them together in maps via MapServer mapfiles. History is maintained for each mapfile, to keep track of changes and, optionally, alter you of such changes.


This is TikiWiki 1.9.3 (CVS) -Sirius- © 2002–2005 by the Tiki community Wed 23 of Nov, 2005 [22:36]
GeoCMS
Tikiwiki Mapserver
English
Tikiwiki GeoCMS
Top Objects
Menu

This homepage is dedicated to the development and use of Mapserver combined with Tikiwiki to create a Geospatial Content Management System (GeoCMS).

We already have lots of installation tips and resources scattered on tw.o and docs.tw.o, but its time to highlight and refactor the most important documentation steps so that more admins may dwelve deeper into open-source geospatial applications.

Figure 2: A map embedded inside a wiki page.

Installation and Setup

Tikiwiki maps feature has been made possible by the publication of the MapServer[73] API for PHP. The MapServer engine is called within Tikiwiki to create images rendered inside the Tikiwiki interface.

I will not spend too much time on installation and setup, as the purpose of this article is to make you want to install it, so I will focus on describing its functionality, however I will quickly summarise.

[73]MapServer web site: http://mapserver.gis.umn.edu/

[74]Tikiwiki installation documents: http://doc.tikiwiki.org/

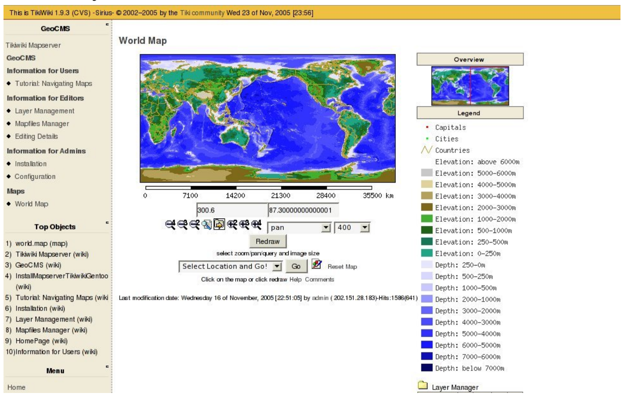
Tikiwiki installation is quite straightforward and a complete installation documentation is available online[74] for various systems including MS-Windows. There are some packages available for various Linux distributions as well. The Tikiwiki installation can be summarised to: untar and fill in a form that directs Tikiwiki to a blank database previously created. The installation of the *php-mapscript* extension from the MapServer code is somehow more complicated as you need to compile *MapServer* with all its dependencies and register the extension with PHP. You will also need to create a directory accessible by Apache and adequately protected to store mapfiles, geographical layers, and generated images. There are also some packages available with some sample data, notably for Mandriva. For this specific installation the documentation can be found here: http://doc.tikiwiki.org/tiki-index.php?page=Maps+Install

Creating maps

Tikiwiki uses the underlying MapServer engine, it means that it follows the mapfile format. This mapfile with the extension *.map* creates a web workspace to be used by the application. A text editor exists in Tikiwiki to edit the mapfile. It also uses quicktags to automatically fill some parts of the mapfile (for instance a template layer). There are several sections or objects in a mapfile. The overall object is the map object and starts with MAP and finishes with END. Inside this object you have the REFERENCE, LEGEND, WEB, SCALEBAR, and QUERYMAP objects. They define the overall representations of the map generated, the overview image (reference), the scalebar, legend and query highlights. Inside the MAP object you can insert LAYER objects. There are mainly two types of LAYERs: raster and vector. The use of the GDAL (pronounced goodle) library for raster and its included OGR (pronounced ogre) for vector allows read and write access to many Geographical Information Systems (GIS) file formats. The most commonly used formats are GeoTIFF for raster and Shapefiles or MapInfo TAB files for vector. Inside the LAYER object you define one or more CLASS objects to specify the color and the type of the feature drawn on the map. You can also add a LABEL object if you would like to label features. A typical mapfile can be found at http://doc.tikiwiki.org/Maps+Mapfile+Tutorial.

Each layer points to a GIS file which is also managed inside a layer management interface, allowing you to upload GIS files to the Tikiwiki site to be used inside mapfiles.

Tikiwiki takes advantage of comments and the METADATA object inside the mapfile. Tikiwiki places a header inside the mapfile, which allows it to keep track of revisions between each mapfile edition, a history is created as well.


This is TikiWiki 1.9.3 (CVS) -Sirius- © 2002–2005 by the Tiki community Wed 23 of Nov, 2005 [23:56]

Last modification date: Wednesday 16 of November, 2005 [22:51:05] by admin (202.151.28.183)-Hits:1586(641)

Figure 3: A table displays the information about the queried objects.

The METADATA keyword WIKI is used to link the layer to a wiki page inside wiki, allowing you to provide explanations on the specific layer. The link is created inside the layer manager on the map interface. The METADATA keyword DOWNLOAD is also used to allow the download of the GIS files composing the layer. This makes Tikiwiki a publication tool of Geographical data.

Finally the METADATA keyword VIEW is used to create pre-defined views of the map, so the user can zoom into the Pacific or Europe, or any other interesting place.

The system permits the inclusion of many layers, but Tikiwiki itself can create some specific layers. For instance, a layer containing the location of all the registered users which have specified a valid location can be created. In the same idea, an image gallery can be declared as geographic type and each image will receive a position. A layer out of all these images can then be created and added to any mapfile as points on the maps. When the user queries such a point, the table containing information about the image and the image itself is displayed just below the map (see fig. 3).

A wiki plugin has been developed which embeds any map inside a wiki page and then performs some basic function like zooming and panning. At its simplest level a generated image can be included inside any web page, including a wiki page, as all the parameters to generate a map image are included inside the URL. GET rather than POST is used to generate maps.

The new viewing interface

I have been working on is *AJAX*-based via the *xajax* library[75] inside the standard viewing interface. But I am also proposing a new interface based on the ka-Map code[76]. I integrated this code inside Tikiwiki and it is provided as a *mods* (module) for Tikiwiki version 1.9.x and 1.10.x. As 1.10.x is still in development, it allows site manager to integrate the new interface immediately by downloading the *mod*.

The ka-Map code has been modified to make it portable, and a small library identifies ka-Map-ready mapfiles by looking for the keyword KAMAP inside the METADATA of the WEB Object. You just specify the name of the map and the various available scales, for instance:

```
KAMAP ?Pacific,1000000,500000,100000?
```

You can also use the standard ka-Map keywords as described in the ka-Map documentation. Finally the code identifies the LAYER METADATA keyword WIKI, and will display it next to the layer legend. The wiki page appears on top of the map with a semi transparent background. This new interface provides a Google Map look and feel for the navigation, while offering customisation of the data sets (see fig. 4).

The future is bright

Tikiwiki as a GeoCMS has been implemented in several countries[77] and is becoming quite successful since MapServer PHP MapScript is quite stable with Apache. MapServer is an Open Geospatial Consortium (OGC) compliant Web Map Server (WMS), which means that many layers can be served over the internet to a wide range of map clients, providing total integration.

On the CMS side, there are several developments being considered. The first one is to include more javascript inside the map interface to show real-time map coordinates at the cursor position, to measure distances and to perform zoom functions based on rectangles drawn on the screen. In general, AJAX could be used to provide an even more user friendly interface.

Some additional functionality will be investigated and developed. For instance, the capability to locate geographical objects by their attributes,e.g.: Where is the town called Suva? For image layers and other layers, the user should be able to hover the cursor over a point to display a box with all the attribute information at this location.

More objects in Tikiwiki will be made geographical, like file galleries, blogs, articles, etc. But the most interesting feature of Tikiwiki is to link trackers with maps. The tracker feature is, in fact, a system to create a form to enter data into a database. For the moment tracker data is only saved in the Tikiwiki database, but through the use of DSN the trackers could be saved in any database, including Postgresql. Postgresql through its PostGIS[78] extension can store geographical objects and MapServer can read such Postgresql databases. Trackers would provide a way to enter any kind of geographical information. The extensions are then limitless including vehicle tracking, fish catch information, soil sampling systems, etc. The future is bright indeed and coders, documenters and users are always welcome.

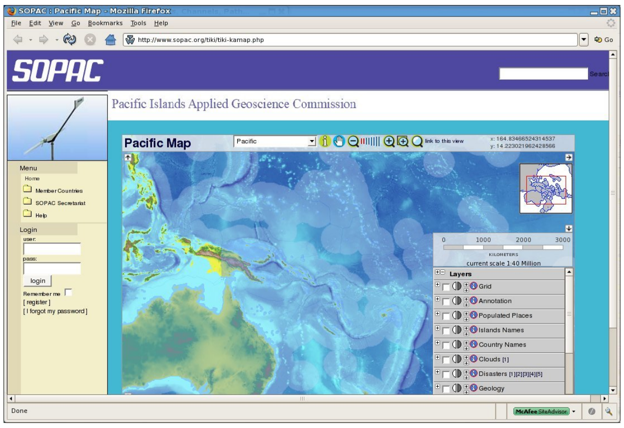


Figure 4: Ka-Map Interface.

[75]XAJAX library site: http://www.xajax.org/
[76]ka-Map site: http://ka-map.maptools.org/
[77]South Pacific countries using GeoCMS: http://www.sopac.org/maps
[78]PostGIS site: http://postgis.refractions.net/

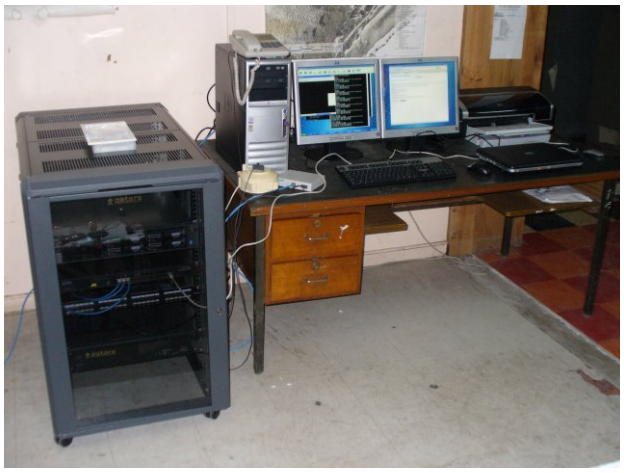


Figure 5: The Mapping Server system in Nauru.

Franck Martin
franck AT sopac.org
http://www.peachymango.org/tiki-index.php?page=Franck+Martin

Franck Martin is lead developer of the maps feature inside Tikiwiki. He works and lives in Fiji in the South Pacific and has been an earlier adopter of GNU/Linux. He is currently working on installing GeoCMS in 14 Pacific Island Countries (see fig. 5). This project is funded by the European Development Fund.

Topical Studies

Spatial Relationships In GIS - An Introduction

Landon Blake (A.K.A. The Sunburned Surveyor)

Introduction

This article provides the reader with an introduction to spatial relationships in Geographic Information Systems (GIS). It presents a simple definition of spatial relationships and explains why spatial relationships are important in a GIS. This article is intended for a reader with a basic knowledge of GIS, and does not cover advanced concepts. To get the most benefit out of this article the reader should [1] understand what a "feature" is in the context of a GIS, [2] understand how features are commonly represented by vector geometry types like points, lines, and polygons, and [3] have a basic knowledge of geometry and coordinate systems.

What are spatial relationships?

In GIS abstractions or simplifications of real world objects are called "features". For example, we might represent a network of roads in an urban area with a network of line segments. Features in a GIS typically have some type of spatial representation. We can use different ways to model the features shape and location in a GIS. Spatial relationships describe how these features are located in relation to each other. By studying the relationship between these features, we can learn more about the real world objects they represent.

Article Scope

In this article we will deal specifically with the spatial relationships between features modeled with "vector geometries". For the purposes of this discussion we define "vector" geometries as shapes that can be expressed using distances, angles, or coordinates. This article does not consider spatial relationships in raster or surface data. Future articles may discuss spatial relationships between raster and surface data.

The shape and location of features in a GIS are commonly described in two dimensions, but may be described in only one dimension, or in three or more dimensions. In this article we will concentrate on the spatial relationships between real world objects in only two dimensions. Future articles may consider the spatial relationships between features represented in three or more dimensions.

Definition of Spatial Relationship

The definition of *spatial relationship* we will use in this article is: "A description of the one or more ways in which the location or shape of a feature is related to the location or shape of another feature."

What Are Vector Geometries?

Vector geometry employs mathematical descriptions of distances and angles or coordinate values to describe the "shape" of a feature. For example, you might represent a single tree as a point, a stream or creek as a line segment or series of connected line segments, and a group of trees or a tree stand as a polygon. In the next section we will consider the ways we can describe the spatial relationships between two or more vector geometries used to represent features in a GIS.

How can we identify and describe spatial relationships?

We can describe the spatial relationship between two features represented by vector geometries in three main ways. You can learn to identify spatial relationships in your GIS data by considering different features and considering how they may be related in these ways.

Three Ways To Identify Spatial Relationships Between Vector Geometries

[1] Compare the measurements of each feature's geometry or shape. [2] Consider the measurements "between" the locations of two or more features geometries and shapes. [3] Consider how one feature touches, overlaps, contains, or connects to another feature. (This is a special type of spatial relationship known as topology.)

In this article we will take a closer look at the first two ways we can describe a spatial relationship. Topology will be discussed in a future article.

An Example

The best way to learn about the first two ways to describe a spatial relationship is with an example. In this example we will look at some of the spatial relationships between two features represented by line segments. A line segment is a portion of a line. In our example we will assume all line segments are straight and not curved. (If we adhere to the strict geometrical definition of a line, a line has no end points and stretches on forever in two directions. That is why we use the term "line segment", which has definite end points. This strict geometrical definition of "line" is not what most lay people think of when they read or hear the word "line".)

Diagram 001 shows two features. The blue feature on the left of the diagram is a creek or a stream represented by a series of connected line segments. The feature on the left of the picture is trail that runs along the creek. The trail is represented by a single straight line segment. I have numbered each point or node at the angle points and end points of the two feature geometries with a 3-digit identifier.

What are some of the spatial relationships between the two features shown in Diagram 001 that fit into the first two categories listed above?

Think about how we can compare the measurements of each feature geometry to the measurements of the other feature geometry. In this case the obvious comparison that we can make is the total or overall length of each feature geometry. (Note that a "feature geometry" is not the same thing as a "feature". A feature represents a real world object, while a feature geometry represents the shape of a feature.) Our creek is longer than the adjacent trail feature, due to its wandering course. This is a relationship that we can quantify using measurement data about the two feature geometries. The total length of the creek between point #101 and point #108 is 370 feet. The length of the trail feature is 345 feet. We know from this information that the creek is 25 feet longer than the trail, and that each foot along the trail corresponds to an average of 1.072 feet of creek.

Now consider how we can describe the spatial relationships between the two features by using the measurements "between" the locations of the feature geometries.

- We can measure the distance from each node on the creek to the trail along a direction perpendicular to the trail.
- We could also compare the angle between each line segment of the creek and the line representing the trail.

- We could compute the area between each line segment of the creek and the line representing the trail.

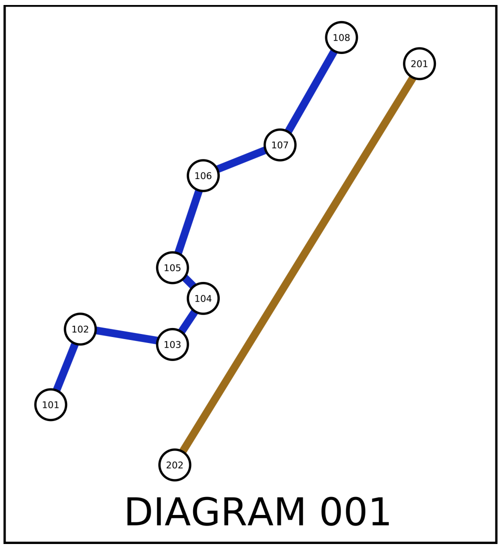


Figure 1: Diagram 001 - An Example

Why are spatial relationships important in a GIS?

When we consider spatial relationships in a GIS they allow us to answer questions about features that we would be unable to answer otherwise. In this way spatial relationships are an aid to spatial analysis. Spatial analysis allows us to answer questions using data that cannot be answered in traditional information management systems such as a relational database. GIS enable this spatial analysis to take place because they track "location" information. You can think of spatial relationships as an important part of the foundation upon which spatial analysis is built. The ability to identify and quantify spatial relationships is consequently very important.

Let's return to the example in Diagram 001 to consider how spatial relationships are an important part of spatial analysis. Think about the type of questions we can answer using the information that we collected about a few of the spatial relationships that exist between the creek and the trail:

- Which portion of the trail is the most susceptible to erosion from the adjacent creek? (This would be portions of the trail located the closest to segments of the creek. It would also be portions of the trail that are more perpendicular to the alignment of the creek.)
- How far along the trail would one have to travel to travel ï£¡ the length of the creek?
- If we are planning to rehabilitate the land between the creek and the trail by planting new vegetation, how many acres of land will we need to cover?

Conclusion

In this article we defined a spatial relationship as "a description of the one or more ways in which the location or shape of a feature is related to the location of shape of another feature." We discussed the three ways you can identify the spatial relationships between two features represented by vector geometries. We can do this [1] by comparing the measurements of one feature's geometry or shape to the geometry or shape of another feature, [2] by considering the measurements "between" the locations of the one feature's geometry or shape to the geometry or shape of another feature, and [3] by considering how the geometry or shape of one feature touches, overlaps, contains, or connects to geometry or shape of another feature. We finished the article by considering a brief explanation of why spatial relationships are an important part of spatial analysis and GIS.

Landon Blake
The SurveyOS Project & The JUMP Pilot Project
http://openjump.blogspot.com/
sunburned.surveyor AT gmail.com

Evaluation of the OGC Web Processing Service for Use in a Client-Side GIS

by Christopher Michael and Daniel P. Ames

Abstract

The Open Geospatial Consortium Web Processing Service proposed specification is intended as a solution for developing web-based geoprocessing plugins, and for easily sharing algorithms and geoprocessing functionality. This paper seeks to evaluate the WPS proposal with respect to feasibility and potential utility, and to identify areas for improvement. Challenges with the WPS proposal are discussed together with potential solutions. Several potential enhancements to the WPS proposal are introduced and considered, including a mechanism to guide client applications in prompting for correct data and a means to list the data available on a server.

Introduction

The Open Geospatial Consortium (OGC, or OpenGIS) is a consensus standards organization concerned primarily with the release of open (i.e., non-proprietary) specifications to unite geographic information software, bringing the multitude of disjoint formats and communications mechanisms together to allow interoperability (Open Geospatial Consortium, 2006). Rather than avoiding this standardization and remaining fully proprietary, many GIS system developers "have shown extraordinary cooperation in teaming to submit OpenGIS Specifications" (Information Today, 1997) and have actively embraced the standards, some even participating in their development. As the OGC is composed of many professionals in multiple fields, rather than a single committee in a single corporate environment, the standards are typically of consistently high quality and are suitable for any number of differing scientific tasks.

On November 17, 2005, the OGC released the fourth revision of a proposal for a specification called the Web Processing Service (WPS) (Open Geospatial Consortium, 2005). This proposed specification describes a mechanism by which geoprocessing may be performed on remote servers, using principally extensible markup language (XML) for communication through the Internet. The specification is authored in such a way that it should be fully language and platform independent. The OGC requested public comments for a time, with a cutoff date of February 4, 2006. Although the forum for public comments has already closed, there have been, to date, few if any real-world studies on the feasibility and utility of the proposal from the client-side GIS point of view.

Prior to WPS, web-based geoprocessing systems and approaches similar to WPS have been implemented by various entities. Most notably, the Environmental Systems Research Institute (ESRI) product ArcInfo 8.3 (ESRI, 2003) 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, where a "Model Builder" tool constructed from simpler ESRI geoprocessing components may be served to ArcGIS Desktop and ArcExplorer clients (Environmental Systems Research Institute, 2006). 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.

What is the Web Processing Service (WPS)?

The Web Processing Service defines a mechanism by which a client may submit a processing task to a server to be completed. The service defines a "server instance", or server, as an entity which may provide one or more processes, or individual processing tasks (e.g., adding two raster datasets together could be one process). In this manner, any given server may be able to perform multiple different, and not necessarily related, processes.

The specification indicates that extensible markup language (XML) should be used for all communication. Extensible markup language documents are made up of individual elements, which are logical containers for related data. An element may contain other elements, and any given element may contain attributes which describe that element. A simple example of an XML document might be:

```
<landscape name="Smithsonian Park">
<tree type="Elm" height="8" />
</landscape>
```

This XML document describes a landscape, which is indicated by the element entitled "landscape". An attribute entitled "name" indicates that this is the Smithsonian Park. A sub-element entitled "Tree" implies that the landscape contains or owns this tree, and the tree has a further two attributes which describe the type of tree and its height. The element is closed, or ended, by repeating the name of the element with a leading slash.

XML is designed to be "straightforwardly usable over the Internet", "human-legible and reasonably clear", "formal and concise", and "easy to create" (W3C, 2000). Using XML is beneficial mainly due to being human-readable, which assists greatly in designing and debugging applications using it. XML documents may be validated to ensure that they contain all needed elements and attributes. Validation takes place against an XML schema, which is a specialized form of XML document which describes the structure that an XML document must follow.

The main goal of the Web Processing Service is to define how to communicate to perform remote processing. To this end, there are three key requests which may be made of a WPS server: GetCapabilities, DescribeProcess, and Execute. The first of the requests asks the server to list the individual processes which are available on that server, along with a short abstract and keywords. The request does not require any parameters. Once a process has been identified from the response, a "DescribeProcess" request may be sent, providing the process in question as a parameter. The response to this request includes the same information as the GetCapabilities response, plus more detailed information about any needed input parameters for the process and whether the input is simple (e.g., a simple number like 23) or complex (e.g., a data file). Complex outputs are typically encoded as XML, for instance using GML (Geographic Markup Language, a relative of XML) for vector data.

If the DescribeProcess response indicated that this is the process the user or client wishes to execute, the third request (Execute) may be invoked. This requests that the server actually performs the operation. Necessary parameters include the name of the process as well as any applicable inputs for the particular process. The response to the Execute request is an ExecuteResponse document, another XML document which indicates a process status, indicates the inputs that were used, and provides either simple literal value outputs or links to complex outputs. The process status may be "ProcessAccepted", indicating that the process was received and is in queue to be processed; "ProcessStarted", indicating that the process is underway; "ProcessSucceeded", meaning the process completed; or "ProcessFailed", indicating that a problem occurred. If the status is ProcessAccepted or ProcessStarted, the status is accompanied by an attribute which indicates where the next ExecuteResponse document may be found. In this way, the client may check on the status of the process by requesting the next ExecuteResponse document. In the case of ProcessStarted, a status message and progress percentage may also be provided.

If the process status is ProcessFailed, the ExecuteResponse document also contains an error code embedded in an XML ExceptionReport element, which may be one of five error codes (MissingParameterValue, InvalidParameterValue, NoApplicableCode, ServerBusy or FileSizeExceeded). If the process succeeded, the response document will also include either the outputs (in the case of simple literal values) or URL links to complex outputs (such as a file with raster data). If a single complex output is produced, that output may be returned directly in lieu of an ExecuteResponse document. Together, these three operation requests and their responses constitute the majority of the Web Processing Service proposal.

Why and when should a Web Processing Service be used?

Because the geoprocessing functionality proposed is unlimited in scope or nature (Open Geospatial Consortium, 2005), the proposal holds great promise for utilizing computational tools without traditional concerns such as distributing bug fixes or checking for the most up-to-date version. Although the scope of what may be accomplished is unlimited, many operations can be completed more quickly locally (i.e., on a user's desktop PC) than remotely (i.e., on a central server), especially after factoring in time to upload input data and subsequently download resulting outputs. When determining whether to use local or remote data processing, a number of factors must be considered beyond the raw size of the datasets involved. Computational complexity plays a large part. If the process takes several hours to complete even on a small dataset, it can be better to process the data remotely. If the task is not complex and the bulk of the work lies in pressing through large volumes of locally stored data, it can be more efficient to perform the processing locally. Hence, before one chooses to use only remote data processing, an intelligent choice ought to be made whether to proceed with processing locally or remotely. As illustrated in the simple graph in Figure 1, remote processing has a useful place in modern computing. Higher processing power in server farms can easily reduce the cost of time-demanding and highly complex tasks, especially when combined with high-throughput networks such as fiber channel or gigabit ethernet. Therefore, data should be sent to servers typically having higher processing power, instead of using the slower local computer, when the time to process the data locally would be greater than the combined time to transmit the data, process it remotely, and download it again.

Remote processing is also an ideal choice if the algorithm is relatively new, and is under active development. In this scenario, new version releases do not require software upgrades by end users. Rather, only the server requires an update to reflect the new code or algorithm. Subsequently all users automatically gain access to the most current and most accurate version of the process simply by using the server-based processing service. This single point of control over the process also creates the possibility of charging for the use of the process, if desired.

The traditional view of remote processing mandates that input data is uploaded, as in Remote Procedure Calls (RPC) where input data and parameters are sent together with a function call (Bloomer, 1992). Input data could be stored on the server as well, requiring the client only to specify the particular input which is desired. This creates the opportunity for a processing server to also provide data: raw data wouldn't necessarily be provided, but processed result data could be returned without needing to upload input data. This makes a great deal of sense particularly for processes requiring real-time data such as weather station observations, live traffic observations, etc. These processes could be provided by the same agency that collects the data, allowing the processes to have access to the latest available data at all times. The motivations for using a remote processing server are many, but ultimately the decision must lie with the user whether remote processing is appropriate for the task.

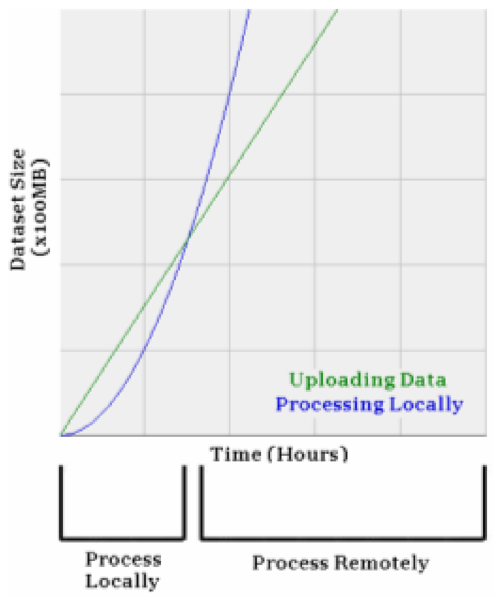


Figure 1: Local processing and remote processing each have their appropriate time.

WPS Implementation Considerations

The WPS proposal describes a mechanism by which a client computer may submit a job to be processed on a server computer. This is classic client/server architecture, meaning that both a client component and a server component are needed. For implementation and testing purposes it is useful to build the client-side component on top of an existing geographic information system to take advantage of existing visualization features for the geospatial data to be processed and returned, however, this is not always required and initial testing may easily be performed with command-line or spatially unaware testing tools. This client-side component is the portion which handles XML communication through the internet with the server, ideally without the user needing to directly see or work with the XML at all in order to discover available processes or to make their request and retrieve results. It is desirable to build this client-side component in such a way that it is not tied to any specific software package, proprietary or otherwise.

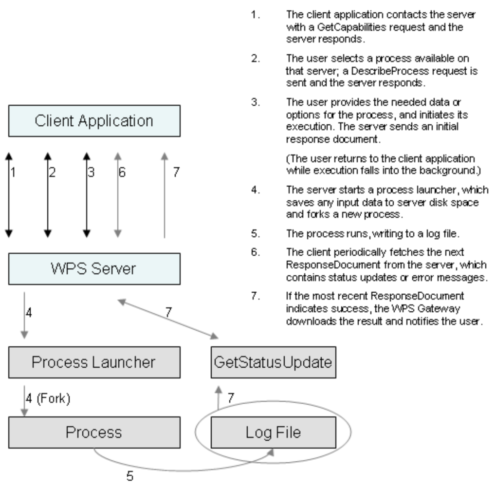
The server-side component which provides the client implementation (and other client implementations) with actual services should similarly be standalone (i.e., not tied to any particular geoprocessing algorithm or process that is to be provided). One approach to this problem is to place an interface layer around existing or new geoprocessing routines written as command-line Linux-based utilities, Windows services, or Windows applications. As long as the application or algorithm implementation in question may be executed without user interaction (e.g., through command-line arguments, TCP/IP communication, or through OLE data transfer), this thin communications layer is a good option. It may be placed around existing geoprocessing functionality and existing tools (or newly developed tools) to enable them to be served using WPS, by providing XML that meets the interface requirements of the WPS communication schema. Rather than requiring a full rewrite of existing software, the existing software will likely only need minor modification to make it able to run in an unattended mode, ideally via command-line arguments and generated log files. GRASS utilities (Neteler, 2006) are an excellent example of existing command-line tools which may be wrapped using an approach such as this.

This wrapper may then initiate the tool as needed, and monitor a log file or a return a status message to indicate whether the process succeeded, is still running, failed or is in queue. Status percentages and relevant error messages may also be retrieved from log files, to be returned to the end user through appropriate communications in an automated fashion, with this wrapping technique. Placing this thin wrapper around standalone tools enables them to be used in multiple places and for multiple purposes with a minimum of effort. In essence, this wrapper becomes the "WPS Server", because it handles all communications needed by WPS. This wrapper, constituting the WPS server, could be implemented as a PHP web page, as an ASP.NET web page, as a standalone application or using any other server technology.

Many of the operations needed by a WPS server are simply metadata operations: providing information about individual processes (i.e., required inputs) and listing processes available on a server. The WPS server will ideally load information on available processes from a configuration file (perhaps one per process) or from a database, thus making the code written for the WPS server reusable simply by adding additional processes to configuration files or to the database. These configuration files or this database may also indicate to the WPS server how to launch the process and how to parse its output files.

An overview of the suggested communications flow between a client application and a server application is shown in Figure 2. Initially, the client application will query the server, providing a GetCapabilities request. The WPS server then returns an XML document matching the WPS schema (arrow 1 in Fig. 2). The client then presents the user with a list of the processes available on the server (arrow 2 in Fig. 2). The user then selects one of these processes, causing the client to request additional information on that process (arrow 3) by sending the DescribeProcess request. The client application helps the user to collect and enter any needed input parameters for the process to be executed; it then initiates the process on the server (arrow 3) by sending the Execute request. The server in turn will trigger a specialized process launcher utility (arrow 4).

Note that the WPS server should not execute the requested process directly, because this would imply a delay: the server would need to complete the process before a reply to the client could be sent. Instead, it is desirable to have the WPS server initiate processing in a different thread, or in a different process, so that an initial ExecuteResponse may be immediately returned to the client. In this manner, geoprocessing operations which are expected to take a good deal of time will still allow the end user to use their computer. For cleanliness and safety, it is best to start a completely new process, so that if a process terminates abnormally it won't stop the entire WPS server as well: hence, a specialized process launcher utility is a good solution.


1. The client application contacts the server with a GetCapabilities request and the server responds.
2. The user selects a process available on that server; a DescribeProcess request is sent and the server responds.
3. The user provides the needed data or options for the process, and initiates its execution. The server sends an initial response document.

   (The user returns to the client application while execution falls into the background.)
4. The server starts a process launcher, which saves any input data to server disk space and forks a new process.
5. The process runs, writing to a log file.
6. The client periodically fetches the next ResponseDocument from the server, which contains status updates or error messages.
7. If the most recent ResponseDocument indicates success, the WPS Gateway downloads the result and notifies the user.

Figure 2: Suggested communications flow between the client and the server.

As explained in section 2, this initial ExecuteResponse document informs the client application of where future status updates (i.e., the next ExecuteResponse) on the requested operation may be found, as well as notifying of success, failure, accepted or rejected jobs, and providing status updates.

The client application should request this next ExecuteResponse document as often as the user wishes (arrow 6 in Fig. 2). Ideally the WPS server should be designed such that an ExecuteResponse document may be found in a consistent location, perhaps always using the same URL for the most recent response. A good approach to addressing generation of response documents would be the creation of a specialized web page or server component which simply parses a log file being written by an executing process to determine current status or errors.

Having the ability to provide status updates requires that the server component is able to get such updates from the process which is executing. As implied, a log file is an excellent solution to this: the process should be able to write a log file or to redirect its standard output to a text file. In this manner, the file may be automatically parsed by the WPS server, providing good connectivity between the components of the system.

Once the ExecuteResponse document indicates that the process has completed, the server must return any applicable error message or return information on where the generated data may be downloaded or obtained. The client software should then allow the user to save or view the data, thus completing the role of WPS.

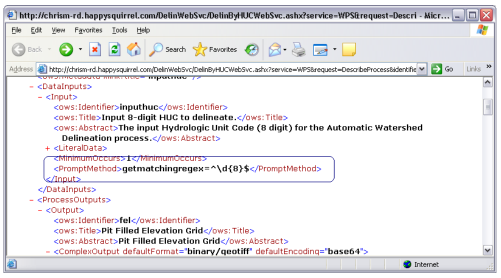
Suggested Enhancements, Problems and Potential Solutions

While the WPS proposal is able to accomplish its stated goals in its current form, the proposal could be enhanced by six key changes. These changes include two additional elements in the DescribeProcess response provided by a server, which describes a given process's inputs and outputs, as well as a mechanism by which a client may cancel a request which is pending or processing. Potential changes also include correcting some inconsistencies in behavior, providing additional exception types for error handling, and having only a single entry point for each process and perhaps a single entry point for each server.

The first suggested change is to add an element to the Extensible Markup Language (XML) document which is returned by a DescribeProcess request. Currently, the needed inputs are listed by this document, but no clear description of how the client should prompt the user for this input is provided. The needed data may come in the form of selecting a shape on a map, providing a literal value such as "23", browsing for a file of a given type, or dozens of other methods for collecting data. In our test implementation, we introduce a new XML element called "PromptMethod" to address this problem. The element may contain the values "browseforvector", "browseforraster", "getboundingbox", or "getmatchingregex". These will cause the client application to prompt for a vector file, prompt for a raster file, retrieve a bounding box (e.g., by asking the user to draw it) or collect a piece of information matching a particular pattern, respectively. The first three require no explanation; the last, getmatchingregex, will accept only a user-entered value which matches the provided regular expression.

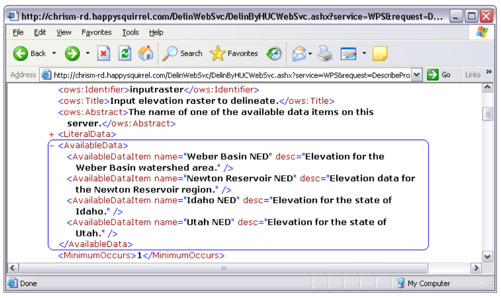
A regular expression is a rule defined by special characters, such as "^[a-zA-Z][a-zA-Z]$". This regular expression would look for the beginning of the input (symbolized by ^), followed by two letters, from A to Z, independent of case, followed by the end of the input (symbolized by a dollar sign). This provides a flexible and quickly configurable input filter to ensure that a user enters only input that is suitable. Some variations for syntax in regular expressions exist due to differing regular expression processing engines (Goyvaerts, 2006), meaning that care should be taken to ensure expressions are designed in such a way that they may be interpreted in the same way on both clients and the server. An example of the suggested addition to WPS may be seen in Figure 3, where the regular expression is defined as "^\d{8}$". This expression indicates that the start of the string (^) should be followed by eight digits (d{8}) and the end of the string ($). This expression assumes that Microsoft's regular expression processor will be utilized.

The second suggested change is another addition to the DescribeProcess response. Currently there is no mechanism by which a server may list the data that is available on the server for use by a given process. An excellent example of where this may be useful is in the processes of watershed delineation - defining stream networks and watersheds based on raster elevation datasets (Savant et al., 2002). Elevation raster datasets can typically be very large, so it is undesirable and often impossible to upload the entire input file to the server. Datasets such as elevation data typically do not change often, and may be stored on the server to save time. In our test implantation, we address this by introducing an XML element entitled "AvailableData", with a child element for each data item containing a name and a brief description as shown in Figure 4. Not only does this speed up processing by removing the need to upload the input data, but it also creates a means by which a WPS server may act as a data repository as well as processing it and returning that processed data.


http://chrism-rd.happysquirrel.com/DelinWebSvc/DelinByHUCWebSvc.ashx?service=WPS&request=Descri - Micr...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://chrism-rd.happysquirrel.com/DelinWebSvc/DelinByHUCWebSvc.ashx?service=WPS&request=DescribeProcess&identifie
Go Links
Internet

```
 - <DataInputs>
  - <Input>
     <ows:Identifier>inputhuc</ows:Identifier>
     <ows:Title>Input 8-digit HUC to delineate.</ows:Title>
     <ows:Abstract>The input Hydrologic Unit Code (8 digit) for the Automatic Watershed
       Delineation process.</ows:Abstract>
   + <LiteralData>
     <MinimumOccurs>1</MinimumOccurs>
     <PromptMethod>getmatchingregex=^\d{8}$</PromptMethod>
    </Input>
   </DataInputs>
 - <ProcessOutputs>
  - <Output>
     <ows:Identifier>fel</ows:Identifier>
     <ows:Title>Pit Filled Elevation Grid</ows:Title>
     <ows:Abstract>Pit Filled Elevation Grid</ows:Abstract>
   - <ComplexOutput defaultFormat="binary/geotiff" defaultEncoding="base64">
```

Figure 3: Suggested PromptMethod element in DescribeProcess response.


http://chrism-rd.happysquirrel.com/DelinWebSvc/DelinByHUCWebSvc.ashx?service=WPS&request=D...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://chrism-rd.happysquirrel.com/DelinWebSvc/DelinByHUCWebSvc.ashx?service=WPS&request=DescribePro
Go Links
Done
My Computer

```
     <ows:Identifier>inputraster</ows:Identifier>
     <ows:Title>Input elevation raster to delineate.</ows:Title>
     <ows:Abstract>The name of one of the available data items on this
       server.</ows:Abstract>
   + <LiteralData>
   - <AvailableData>
       <AvailableDataItem name="Weber Basin NED" desc="Elevation for the
         Weber Basin watershed area." />
       <AvailableDataItem name="Newton Reservoir NED" desc="Elevation data for
         the Newton Reservoir region." />
       <AvailableDataItem name="Idaho NED" desc="Elevation for the state of
         Idaho." />
       <AvailableDataItem name="Utah NED" desc="Elevation for the state of
         Utah." />
     </AvailableData>
     <MinimumOccurs>1</MinimumOccurs>
```

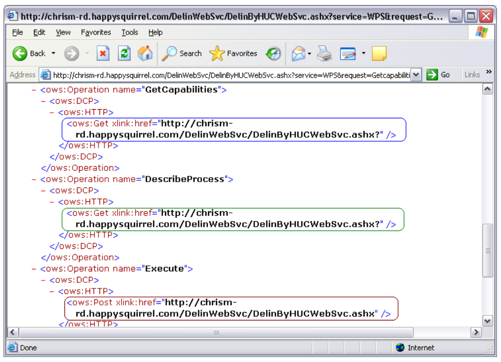
Figure 4: Suggested AvailableData element in DescribeProcess response.

In the current WPS proposal there is an inconsistency after submitting a job to a server with regards to what should occur next.. If the process will result in a single return value and the Execute request was made with the "store" parameter set to false, WPS will allow the process to immediately return the output, rather than an XML ExecuteResponse document. If there is more than one output, or if the process has been asked to store the results, then an ExecuteResponse document is generated. This behavior is inconsistent, since any given process might return either multiple outputs or a single output, depending on the parameters provided to the process. Instead, it is simpler and more straightforward to always return an ExecuteResponse document, storing any complex output data (e.g., vector or raster files) on the server until downloaded by the client. Simple value outputs (e.g., a single number) may be returned directly embedded in the ExecuteResponse document, with links pointing to complex outputs. Our third suggested change is to require always returning an ExecuteResponse document to provide greater consistency and simplify client implementation far easier.

After submitting a job to a WPS server, it may be desirable to cancel the requested operation; this eventuality is not planned for with the current WPS proposal. In our test implementation (and as our fourth suggested change) we introduce the use of a "cancel request URL" in the ExecuteResponse document, along with the existing URL indicating where the next ExecuteResponse document may be found. In this manner a client wishing to cancel a process needs only to access the URL to trigger a cancellation of the requested process, preventing wasted server processing time on undesired processing.

Our fifth suggested change to the WPS proposal is to have a single entry point for each possible request (GetCapabilities, DescribeProcess, and Execute) on a given process. Ideally, a single entry point (e.g., a single PHP web page) could be used not only for every request on a process, but also for every process available on that server. Presently, each of the requests that a process supports may have a different URL to perform that request, as illustrated in Figure 5. In the figure, the GetCapabilities URL is circled in blue, the DescribeProcess URL in green, and the Execute URL in red. Here each URL is the same, making maintenance and implementation easier, although it is acceptable to have a different access URL for each operation. Confusion or errors can easily arise with differing URLs for these operations. This can be addressed easily by making it required to use only one URL for all three of the requests that a process must support.

Lastly, our sixth suggested change is to implement a more highly structured exception system. Presently there are only a few exception types (MissingParameterValue, InvalidParameterValue, NoApplicableCode, ServerBusy and FileSizeExceeded), which are limiting - particularly when attempting to parse the error automatically. With a small number of exception types (exception codes), there is no generic way in which to handle errors on behalf of the user; the only thing that can be done presently is to display the error to the user. A more complex and structured exception hierarchy would allow client applications to understand the nature of the error that occurred, perhaps recovering or retrying automatically as needed. The ability to insulate the end user from errors and recover gracefully is an important part of any standard design, and this ability would be made possible with a more detailed exception hierarchy.


http://chrism-rd.happysquirrel.com/DelinWebSvc/DelinByHUCWebSvc.ashx?service=WPS&request=G...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://chrism-rd.happysquirrel.com/DelinWebSvc/DelinByHUCWebSvc.ashx?service=WPS&request=Getcapabiliti
Go Links
Done
Internet

```
- <ows:Operation name="GetCapabilities">
  - <ows:DCP>
    - <ows:HTTP>
        <ows:Get xlink:href="http://chrism-
          rd.happysquirrel.com/DelinWebSvc/DelinByHUCWebSvc.ashx?" />
      </ows:HTTP>
    </ows:DCP>
  </ows:Operation>
- <ows:Operation name="DescribeProcess">
  - <ows:DCP>
    - <ows:HTTP>
        <ows:Get xlink:href="http://chrism-
          rd.happysquirrel.com/DelinWebSvc/DelinByHUCWebSvc.ashx?" />
      </ows:HTTP>
    </ows:DCP>
  </ows:Operation>
- <ows:Operation name="Execute">
  - <ows:DCP>
    - <ows:HTTP>
        <ows:Post xlink:href="http://chrism-
          rd.happysquirrel.com/DelinWebSvc/DelinByHUCWebSvc.ashx" />
      </ows:HTTP>
```

Figure 5: Ability to have a differing URL for each request on the same process can cause confusion and unnecessary complexity.

Conclusions

Overall, the WPS proposal was found to be an effective way of standardizing on a communications mechanism for client/server geoprocessing. The proposal works as currently designed, and is indeed suitable for many GIS tasks. Implementation of the client component and the server-based WPS processes showed that the WPS proposal is sound and effective. Nonetheless, there are several opportunities for enhancement to the WPS proposal including the six specific recommendations provided here. It is hoped that these observations will benefit others working to implement the WPS proposed specification on various GIS platforms.

Any conceivable geoprocessing activity may be presented in a WPS format. This allows the developers of the algorithm to continuously improve the algorithm while still allowing users at any location to use the code, without needing to update to a latest version. This also ensures that code and algorithms being used are of high quality; if a bug is detected in a released version of an algorithm which is distributed on CD, no easy way exists to ensure that all users successfully upgrade to the corrected version. With a web-based processing architecture, this problem is alleviated by simply publishing the new algorithm to the processing server. Processor intensive and time consuming geoprocessing may be performed on remote servers, freeing the user's desktop to work on other tasks. Large datasets may even be stored on the server and processed according to the user's particular parameters, combining data repository and data processing aspects into one system.

The WPS proposal introduces a standard which will allow diverse developers at different locations to produce geoprocessing offerings and provide them easily to differing client platforms. In these tests, the WPS proposal has been shown to be practical and usable in its current design. The additional enhancements proposed can improve WPS significantly, making it better suited still to the task for which it was designed.

Bibliography

[1] Bloomer, John. 1992. **Power Programming with RPC. Cambridge, MA:** O'Reilly Media.

[2] Environmental Systems Research Institute (ESRI). 2003. **ArcGIS Desktop Products Data Sheet.** WWW document, http://www.esrichina-bj.cn/produce/esri/arcgisdesktopsheet.pdf

[3] Environmental Systems Research Institute (ESRI). (2006). **ArcGIS 9.2 Webinar – ArcGIS Server: Publishing a Geoprocessing Model.** WWW document, http://events.esri.com/info/index.cfm?fuseaction=seminarRegForm\&shownumber=9919

[4] **GIS Competitors Cooperate on OpenGIS Specs. 1997.** Information Today, 14(2), 15-15.

[5] Goyvaerts, Jan. 2006. **Regular Expression Tutorial.** WWW document, http://www.regular-expressions.info/tutorial.html

[6] Open Geospatial Consortium, Inc. 2006. **Vision and Mission.** WWW document, http://www.opengeospatial.org/about/?page=vision

[7] Open Geospatial Consortium, Inc. 2005. **OpenGIS® Web Processing Service (WPS) Discussion Paper.** WWW document, http://www.opengeospatial.org/

Christopher Michael
Research Assistant
Idaho State University Geospatial Software Lab

Daniel P. Ames, PhD
Assistant Professor
Department of Geosciences, Idaho State University

Developer Announcements

Developer Updates

For First Quarter of 2007

Omniverdi Livecd

by Luca Casagrande

What is a LiveCd

As Wikipedia reports, Livecd is a "generic term for an operating system distribution that is executed upon boot, without installation on a hard drive. Typically, it is stored on bootable media such as a CD-ROM (Live CD), DVD (Live DVD), USB flash drive (Live USB), among others." By using this technology, users can get a "taste" of a full GNU/Linux box without fighting with any setup or installation. Once the LiveCd has been booted, everything is loaded in RAM, keeping the hard disk safe. Of course, once the machine shuts down, every change is lost.

Applications

There are many applications that can be satisfied through the use of a livecd:

1. Running your favorite system anywhere you can find a PC. With a Pen Drive to save your data, you will have a perfect survival kit.
2. For teaching purposes: During a workshop, your audience can have anything you think is appropriate to increase the learning process.
3. Using an old machine: liveCd can be run on an old machine and also without a HD!
4. Spreading data: it's a very cool way to distribute your work. You deliver your work and all the applications that are needed and configured to appreciate it best! For example, delivery of a comprehensive solution including all data and applications.

Limitations of a Livecd

Inside a livecd we have the kernel, drivers, WM, and all the applications required for a working environment. Of course, it is rather impossible to put all the drivers that would suit all the machines that have been created. It can happen that, during boot, the CD stops or something else goes wrong (e.g. missing lan detection). Thanks to the work of a lot of people (in primis Knoppix) we have very good software that detects hardware during boot but, like every software application, there can still be bugs. As you can understand, the idea of having a universal machine is a high expectation and the road to achieving this is still long.

On the other hand, no information is saved while working in the LiveCd environment. If this is a favorable aspect (e.g. keeping your hard disk safe), it can be tedious from another point of view. A solution is to use a Storage Device (like a USB HD, for example) to store all your output. This approach can also be used to save configuration files, so that you can keep your settings after rebooting.

LiveCd and open source geospatial software

LiveCd can be a valid tool if applied to open source geospatial software.

An example of an application of liveCd and GIS FLOSS is to run a workstation that will allow users to explore data inside it or, for example, in a remote Postgis DB. Old machines can be used to make access points to the database and can be set up with a viewer that is always available (as Internet Points do with a browser), allowing the saving of data on a storage device.

During the FOSS4G 2007 at Lausanne, we used 2 LiveCds during seminars (ka-Map and Grass 3d). We also allowed participants to immediately test what they had just see in the slideshow. Once the workshop ended, they were able to keep their liveCd and use it at home or at their office. This is a good example of the great potential of a liveCd deployed for Educational purposes.

Objective

Our liveCd will always try to offer users the following:

1. Up to date software with every main official release
2. Included guides, documents and tutorials
3. Good hardware compatibility

Technical development

We decided to use Catalist: the same tool that the Gentoo Release Engineering Team uses.

This choice is based on 2 main reasons:

1. To use a system that will keep the updating stage fast and safe
2. To allow the customization of everything inside the CD

Desktop 2007.0

Our first release for 2007 contains this software:

- QGIS 0.8
- GRASS 6.2.1
- PostgreSQL 8.1.5
- PostGIS 1.1.4
- GDAL 1.3.2

The system runs a 2.6.18 Linux kernel and uses GNOME as a Desktop Manager.

While writing this abstract, we are waiting for the release of the 2007.1 Qgis 0.8.1.

Future Development and Tasks

Our goal is to keep the CD updated by creating new versions on a regular schedule, or at least as soon as major updates or included software will be released.

At the moment we are working on two tasks:

1. An Installer for the livecd
2. A server edition with mapserver and some webgis applications

LiveCD installer

The installer will be accessible through an icon on the desktop. The installation process will guide the user to the setup of the GNU/Linux distribution on his/her computer. Hard disk installations will allow better performance and the possibility to update all packages, thanks to the core of the Gentoo Linux (http://www.gentoo.org/)system.

Server Edition

The server edition will target a server machine with minimal graphical support but with a fully-featured set of applications to offer OCG Web Services and UMN Mapserver frontends (ka-Map, Open Layers, p.Mapper, ecc.).

[79] LiveCD: http://livecd.ominiverdi.org

[80] http://www.gentoo.org/proj/en/releng/catalyst/

More Info

- Ominiverdi Livecd Project [79]
- Gentoo Catalyst Project [80]

GDAL Status Update

by Frank Warmerdam

1.4.0 and 1.4.1 Releases

The year 2007 started with the GDAL/OGR 1.4.0 release. This release included new raster drivers for WCS (Web Coverage Server protocol), PDS and ISIS (mars). It included new vector drivers for Informix, KML and E00 formats and many feature improvements to the existing drivers.

By the time you read this, the 1.4.1 release should be in general availability, and it includes on the order of 40 bug fixes. The 1.4.1 release is the first GDAL stable release. That is, it is the first time the GDAL project has split off a stable branch and focused only on bug fixes in that stable release branch, while new development goes on seperately. This should make it possible to produce bug fix releases on short notice without the risk of introducing new bugs due to new development.

Infrastructure

The last few months have also seen changes in several aspects of GDAL/OGR infrastructure.

- Migration from the CVS to the Subversion source control system, with hosting now at OSGeo.
- Migration from the Bugzilla to the Trac bug tracking system, with hosting now at OSGeo.
- Introduction of wiki facilities for the project via Trac.
- Migration of the web site, and download facilities to OSGeo / Telascience servers, from my personal server.
- Use of central LDAP authentication for Subversion and Trac shared with other OSGeo projects.

I would especially like to thank Howard Butler who was intrumental in making the Subversion and Trac migration painless while preserving the nearly decade of history in the old systems.

Sponsorship

The last few months have also seen the launch of the GDAL/OGR Sponsorship program operating through OSGeo. Our first sponsors are Analytical Graphics, Applied Coherent Technology, SRC, Safe Software, Cadcorp, Waypoint and MicroImages.

With the generous financial support of our sponsors the project has been able to contract Mateusz Loskot to perform general maintenance and support tasks for the GDAL/OGR project from March to August. With the help of Mateusz and other members of the GDAL/OGR development team, the 1.4.1 release has seen a record number of old bug reports cleaned up. Contracting Mateusz has put the project in a stronger position to get needed tasks done regardless of whether it is a current itch for any of the developers.

Many thanks to Mateusz and our sponsors! I believe this will be a model for sustaining needed work on many OSGeo projects. For information on our sponsors, and the GDAL/OGR sponsorship program, visit: `http://www.gdal.org/credits.html`

GeoNetwork Open Source

by Jeroen Ticheler

Introduction

Work has been concentrated on a couple of aspects of the project lately. They are:

- Development of GeoNetwork opensource version 2.1 resulting in beta releases
- Upgrading of the Community website
- Work related to the OSGEO incubation process

Development of version 2.1

A lot of development has been going over the last 9 months. Many new functions have been added and existing ones have been enhanced. The plan is to release a final version 2.1 of the catalog software towards the end of April 2007. Here's a summary of what has been developed for this release:

- Support for the ISO-19115 metadata standard, formatted and validated against the ISO-19139 implementation schema. Read more and also refer to the ISO19115 / 19139 confusion for more details on those standards.

- Support for the OGC Catalog Services for the Web (OGC-CSW 2.0.1) base specification, based on the ISO application profile. Read more
- Web based harvesting configuration, scheduling and monitoring between GeoNetwork opensource nodes and from Web accessible folders Read more
- Embedded thesaurus support, both in the metadata editor and in the search
- Simplified installation, upgrade and migration with functionality moved to the Web based configuration and to the stand alone GeoNetwork Administration Survival Tool (GAST)
- Improved multi-lingual support with online translation functionality for categories, regions etc...

Upgrading of the Community website

The community website has been moved to a telascience server with the assistance of John Graham. The focus of the website will be on the users of the GeoNetwork opensource software. More work has to be done to clean up the content and remove developer related content.

The Plone based website was upgraded and the content moved. The site now allows for multilingual content, PDF output of pages and a cleaner structure. A Documentation Center and a Download center form the basis of the website.

For developers, a Trac-based site `http://trac.osgeo.org/geonetwork` will be the central website, providing a bug track mechanism and a WIKI. The intention is to also move the SVN repository to this server to benefit from trac-svn integration. Developer related content from the Community website has to be moved here.

OSGEO incubation process

The project is working on the incubation process to OSGEO. The code review has been completed, web pages have been updated to reflect the OSGEO connection. A review of the software's dependencies is about to be completed without foreseen problems.

GeoTools Steering

This report is written in the first quarter of 2007, and outlines community development and research goals

by Jody Garnett

Users Community

GeoTools has started up production of monthly milestone releases again. The publication of milestone releases makes progress visible and accessible to the user community. We have not been sending announcements out beyond the user list, as milestone releases are strictly for "early access".

- 2.4-M0
- 2.4-M1

We have also updated our Module Matrix page (and development policies) to reflect how well different modules are supported.

Development

This time around we are balancing between several paid projects (with deadlines) and Martin's presentation to the GeoAPI working group (summer OGC meeting?).

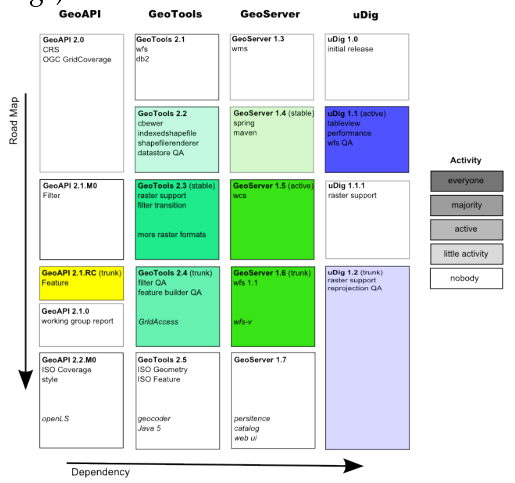

GeoAPI
GeoTools
GeoServer
uDig
Road Map
GeoAPI 2.0 CRS OGC GridCoverage
GeoTools 2.1 wfs db2
GeoServer 1.3 wms
uDig 1.0 initial release
GeoTools 2.2 cbewer indexedshapefile shapefilerenderer datastore QA
GeoServer 1.4 (stable) spring maven
uDig 1.1 (active) tableview performance wfs QA
GeoAPI 2.1.M0 Filter
GeoTools 2.3 (stable) raster support filter transition more raster formats
GeoServer 1.5 (active) wcs
uDig 1.1.1 raster support
GeoAPI 2.1.RC (trunk) Feature
GeoAPI 2.1.0 working group report
GeoTools 2.4 (trunk) filter QA feature builder QA GridAccess
GeoServer 1.6 (trunk) wfs 1.1 wfs-v
uDig 1.2 (trunk) raster support reprojection QA
GeoAPI 2.2.M0 ISO Coverage style openLS
GeoTools 2.5 ISO Geometry ISO Feature geocoder Java 5
GeoServer 1.7 persitence catalog web ui
Activity
everyone
majority
active
little activity
nobody
Dependency

Figure 1: The above diagram is color coded based on project and developer activity (the amount of colour indicates the rate of change).

Development Policies

Since 2006 Q4 the GeoTools project and GeoServer projects have made development policy changes:

- GEOT:GeoTools change proposal - Changing an existing API
- GEOT:Supporting your module - Adding a new module to GeoTools
- GEOS:GSIP 6 - Track GeoTools Trunk

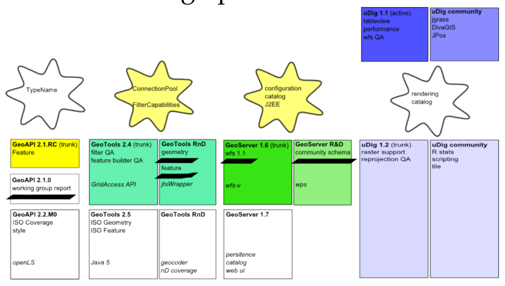
The above policies aim to have a uDig and GeoServer branch available based on GeoTools trunk, without this support we are left with a development community scattered across different versions (all trying to be active).

There is one planned development policy change:

- When a contributor agreement is available (after negotiations with OSGeo) we will require each developer to sign something.

Research Goals

There are several tracks of ongoing research, and a few unresolved design problems:



Research Policies

The GeoTools project and geoserver project made policy changes with respect to new research.

- GEOT:Creating your own Module

The establishment of an "unsupported" module space in GeoTools has succeeded in making visible research that was previously "out of sight out of mind". The uDig and GeoServer projects already and community sections meeting this need.

Known Deadlines

In the above diagram the following known deadlines are underlined.

Here are the dates as near as have been made public:

- GeoAPI Report for June 11th (May 11th code freeze)
  - Report to be reviewed during July 9th Meeting working group meeting
  - The report must be submitted for June 11th (ie one month before)
  - Last time this report took a month to prepair so assume a code freeze on May 11th
- GeoTools ISO Geometry implementation for Mid March
  - Either jtsWrapper or geometry to be brought up to supported status
  - Module may not be included in download until Java 5 is available

The following are research goals only (no formal release required):

- GeoTools Feature / GeoServer Community schema for Mid March
  - the implementation is being made available as an unsupported module, actual intergration happens later

Outstanding Problems

The following problems are listed as design problems that should be addressed. Often these problems are holding up existing paid work, but due to the amount of collaboration involved lack specific funding.

- **GeoAPI TypeName** - We are caught between the ISO definition of GenericName (in which the Schema in which the name needs to be known prior to construction) and the need to have a quick "id" when looking up feature content. This is an example of a usibility tradeoff .. hard to resolve. The Java class QName is an example of a good compromise (based on the same ISO specification).
  - GeoAPI TypeName interface has been defined in a manner similar to QName, this is in conflict with the interface for GenericName
- **GeoTools ConnectionPool** - GeoTools is starting to be used in more serious J2EE applications, as such we need to use a JNDI look up for our DataSource (Use of an oracle ConnectionPool is most often requested on the user list).
  - We need to define an approach, document it, and roll it out into the database epsg authorty plugins and datastores.
- **GeoTools FilterCapabilities** - We have switch over to use GeoAPI Filter interfaces. The interface "Function" does not list the number of required parameters; instead that information is captured as part of the FilterCapabilities information (which we currently do not use).
  - We need to split out the description of avaialble functions from the data structure used to define an expression.

- **GeoServer Persistence** - The ability to persist GeoServers state (currently to an XML file) is hard to maintain, and not extensible.
  - We have several sources of inspiration (using an XML to bean technology), some restrictions (allowing for programatic configuration) and some acceptence tests (ability to add additional configuration elements without changing the object model).
- **GeoServer Catalog** - A catalog is used to manage your resource connections, and allow for additional kinds of resources over time. The existing approach (cut & paste) is not scaling well for GeoServer.
  - We need to bring the GeoTools interfaces up to the level in which they are used by GeoServer and uDig. There are lots of negative examples (on what we don't want to do) and a few good ideas. GeoTools should strive for the minimum common ground.
- **GeoServer J2EE** - GeoServer needs to become a better J2EE application
  - Pay attention to DataSources retrived via JNDI lookup
  - support external configuration and clustering
- **uDig Rendering** - The uDig rendering system has seen enough real world use that we can start to simplify based on experience. Depending on where the line is drawn in the sand we may be able to add OpenGL to our list of targets (currently SWT and AWT are supported).
  - While the work can be backported to GeoTools we need to ensure that there is a good reason (measured in developers) to make the effort worthwhile.
- **uDig Catalog** - The uDig catalog system has seen enough real world use that we can start to simplify based on experience.
  - While the work can be backported to GeoTools we need to ensure that there is a good reason (measured in developers) to make the effort worthwhile.

GRASS GIS

by Markus Neteler

About GRASS

GRASS (the Geographic Resources Analysis Support System) is a vector and raster GIS, image processing system, graphics production system, and spatial modelling system. It contains many modules for raster and vector data manipulation, rendering images on the monitor or paper, multispectral image geocoding and processing, and attribute management. It is published under the GNU General Public License. GRASS is written in the C language with a prototype GRASS-SWIG interface and further Python based WebGIS applications. The data management capabilities also include 3D raster (voxel) modelling as well as vector network analysis. Through the R interface geostatistical analysis can be performed. GRASS is portable multi-platform software running on Linux, MacOS X, Posix compliant Unixes and MS-Windows (through Cygwin, and major parts now also working natively).

Recent Events

- Spring 2006: GRASS Quality Assessment (QA) source code navigator online
- 22 Feb 2006: GRASS GIS 6.0.2 released - bugfix release
- 12 April 2006: Experimental native winGRASS 6 version integrated with QGIS
- 11 August 2006: GRASS 6.1.0 released - this is a technology preview release
- 31 Oct 2006: GRASS 6.2.0 released - The stable version is published
- 12 Dec 2006: GRASS 6.2.1 released - This release fixes several bugs discovered in the 6.2.0 source code
- 20 Dec 2006: GRASS GIS / OSGeo Newsletter Published - The first combined GRASS-News / OSGeo-News volume is available
- 10 Feb 2007: GRASS GIS 6.2.1 winGRASS/Cygwin binaries available
- 12 Feb 2007: New GRASS bug and wish tracker - Gforge based
- Italian GRASS and GFOSS Users Meeting - GRASS and GFOSS Users Meeting, Palermo (Italy), 14-16 Feb 2007

Under Development

Besides hard testing and bugfixing, several major projects are underway:

- native winGRASS port to be completed

- GRASS GIS 6.3.0 to be published (draft announcement)
- new Python based graphical user interface in progress
- Work related to the OSGeo incubation process (code vetting and addition of missing copyright statements)

Statistics

- In the first three months of 2007, the main server statistics show (there are more than 20 mirror sites, not included here):
  - more than 5000 GRASS 6.2.1 source code downloads
  - more than 2000 MS-Windows/Cygwin downloads
  - more than 1000 Linux downloads
  - 11100 Spearfish sample dataset downloads
- grassuser mailing list
  - more than 800 members
  - averaging about 15 messages/day
- grass-dev mailing list
  - more than 450 members
  - averaging about 20 messages/day
- twelve additional mailing lists exist plus several national GRASS user lists
- Over the last 8 months, every 1.16 hours a change is committed to the GRASS source code repository (source)
- More useful stats can be found on the Ohloh site: `http://www.ohloh.net/projects/3666`

Mapbender Project Update

by Christoph Baudson

About MapBender

MapBender is the software and portal site for geodata management of OGC OWS architectures. The software provides web technology for managing spatial data services implemented in PHP, JavaScript and XML. It provides a data model and interfaces for displaying, navigating and querying OGC compliant map services. The Mapbender framework furthermore provides authentication and authorization services, OWS proxy functionality, management interfaces for user, group and service administration in WebGIS projects.

Recent Events

January 2007

- Mapbender 2.4.1 rc1
- adoption of coding conventions
- establishment of module maintainers

February 2007

- new issue tracker
- establishment of a release owner

March 2007

- weekly developer IRC meeting
- Mapbender 2.4.1
- face-to-face developer meeting in Berlin

New Mapbender features

- **New administration tools**: metadata handling, new administration interfaces.
- **New treefolder module**: enhanced usability by centralizing controls in one module. Controls in pop-up windows are now obsolete. Also features a WMS status indicator.
- **Metadata gazetteer**: a text search in WMS/layer metadata and keywords. Retrieves WMS or single layers by an AJAX-query.
- **Improved WFS-T configurability**: enhanced user interface for easier and richer manipulation of WFS-T data.
- **Improved digitizing module**: a completely new design following an AJAX-approach. Improved usability and performance.

Future Development

- **Integration of OpenLayers**: an integration of OpenLayers as an optional Mapbender module is scheduled for 2007.
- **WFS-T / Digitizing enhancement**: handling of complex geometries.
- **wider use of AJAX /JSON and OO technology**: improvement of module interfaces to do justice to wider development audience.
- **evaluation of packing algorithms**: due to traffic minimization, there are few source code comments. JSDoc and PHPDoc will be used to comment in the future. The traffic issue will be

resolved by extracting the comments prior to deployment and by packing JS-code.

Statistics

- **Increase in developer mailing list participation**: 2006 Q3: 40 posts, 2006 Q4: 121 posts, 2007 Q1: 209 posts (until March 19)
  Take a look at Mapbender statistics at Ohloh. Here are some excerpts:
- **Project worth**: $2,436,378. "This calculator estimates how much it would cost to hire a team to write this project from scratch."
- **17 developers**: "Over the past twelve months, 10 developers contributed new code to Mapbender. This is a relatively large team, putting this project among the top 5% of all project teams on Ohloh. For this measurement, Ohloh considered only recent changes to the code. Over the entire history of the project, 17 developers have contributed."
- **Increase in activity**: "Over the last twelve months, Mapbender has seen a substantial increase in activity. This is probably good sign that interest in this project is rising, and that the open source community has embraced this project."

MapBuilder Project Update

Strategic Direction March 2007 *by Cameron Shorter*

Status

MapBuilder is a powerful, standards compliant and FREE geographic mapping client which runs in a web browser. It renders raster maps from WMS, Google Maps and more, and vector layers from WFS, GeoRSS and GML. It even offers feature editing to WFS-T. Mapbuilder is often described as a web based toolkit, or framework, that allows a developer to insert a selection of widgets into a web page. Eg: MapPanes, FeatureLists, Navigation tools, Style Editors and more.

Over the last year, the three leading browser based mapping clients, Mapbuilder, OpenLayers and MapBender have been actively working together sharing ideas and code. In particular, OpenLayers is currently being inserted into Mapbuilder as a rendering engine. Our latest release, mapbuilder-1.5apha1 includes an OpenLayers renderer and our next release will complete the integration by linking Mapbuilder and Openlayers tools.

OpenLayers focuses on rendering a MapPane. Mapbuilder extends this to offer extra widgets like Style Layer Descriptor (SLD) Editors, Time Series Web Map Services, processing Web Map Context (WMC) documents and more.

Mapbuilder developers are now contributing to the OpenLayers codebase when adding core Mapping functionality.

Merging code between projects is difficult emotionally as much as technically. The catalyst for the Mapbuilder/OpenLayers merge was that there were four different projects developing vector rendering using SVG/VML at the end of 2006. After much discussion we agreed to work together on the same code base. This meant that each of us had to throw away 3̃/4 of our original code. In the short term, this meant some extra effort from all of us, but in the long term, we will all benefit from the merger. We will have more developers maintaining the same code base and users will be less confused when trying to pick a client.

Developers who throw away code feel a strong sense of loss of status and credibility. Credit goes to those who were flexible enough to throw away their code and also to the developers who were generous in their acknowledgments of past works.

Internally, Mapbuilder stores its map data inside a Context document. Initially we used to use a Web Map Context (WMC) document which describes a list of WMS layers. However, the WMC doesn't allow you to insert other layers, like WFS, GML, GeoRSS, Google Maps, etc.

In mapbuilder-1.5alpha1 we support OWS Context (currently an OGC draft document) which extends WMC to include multiple layers types. This improves the structure of our code and configuration files, as well as continuing with our support of OGC Standards. The 1.5 branch will stablise over the next few months with release candidates and then final release.

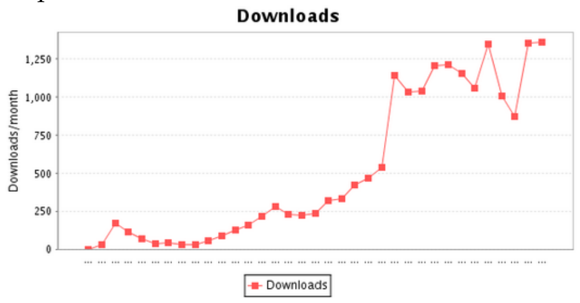
Mapbuilder graduated through the Open Source Geospatial Foundation (OSGeo) incubation process in October 2006 making the second and most recent project to graduate. Graduation turned out to be a lengthy process involving auditing code and tidying up development processes. However compared to other projects going through graduation, Mapbuilder is young and had less history to dig through, which is why were one of the first projects to graduate (after Mapbender).

Association with OSGeo was a positive move for Mapbuilder popularity. Mapbuilder downloads doubled in the month after OSGeo was created with Mapbuilder as one of the founding projects.

In summary, Mapbuilder continues to have a healthy developer and user base, it has a stable code-base, good development processes and a healthy future ahead of it.

Download Metrics

This graph shows a steady growth in interest since the start of the project, with a doubling of downloads when the OSGeo Foundation was founded (with Mapbuilder a founding project) in March 2006. There was a dip over Christmas 2006, then back again early 2007 (the gml-viewer and mapbuilder-1.5alpha1 was released at the end of December 2006).



Subversion Commit Metrics

This graph tracks the number of lines of code in the Mapbuilder repository. The graph shows a that mapbuilder has been steadilly growing since December 2003. We have had consistent support from a community of developers which includes a constant core, as well as a number of fring developers who join the community, develop for a while, then move on.

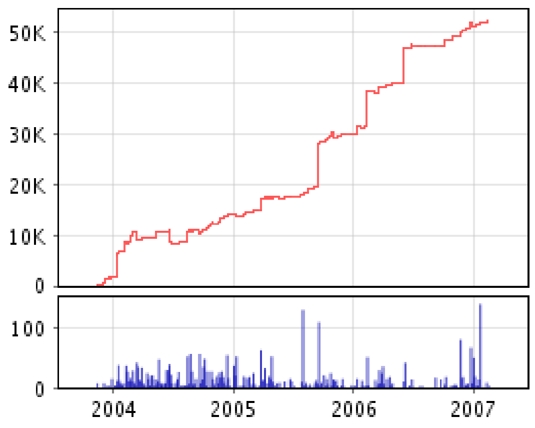
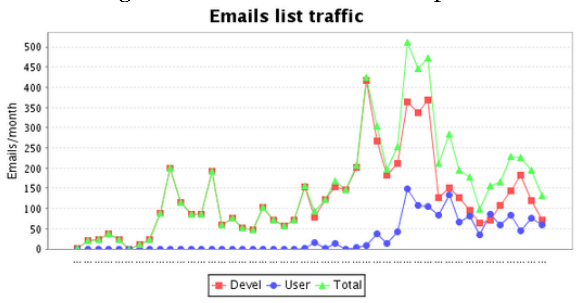


Figure 2: Source data http://fisheye.codehaus.org/browse/mapbuilder/trunk/mapbuilder/mapbuilder/lib

Email Traffic Metrics

This graph shows the monthly number of emails for Developer and User email lists. Developer emails spiked at the end of 2005, beginning of 2006 during the development of release 1.0 before leveling back. The stabling of email traffic can be put down to:



1. A reasonable effort has been put into documentation. Often newbie questions are simply answered by pointing at existing documentation. This is not to say our documentation is great, it is just getting a bit better.
2. The Vector rendering work done from mid 2006 was done in conjunction with outside developers. Much of the communication moved to other email lists, or IRC channels. (In particular, irc://freenode.net#openlayers
3. April - October 2006, a reasonable amount of developer effort was focused on getting OSGeo Graduation rather than development. A drop in email activity over the last few months is probably due to:

Gml Viewer Download Metrics

In December 2006, we released GML Viewer version of Mapbuilder which can be distributed with a GML Dataset. It was distributed on CD as one of the artifacts of the Open Geospatial Consortium Testbed 4. It was also made available for download and download statistics are available below. The number of downloads is 1/10 of the mapbuilder-lib downloads above which suggests that the target market are getting their gml viewers from the CD, or that people are more interested in other features from Mapbuilder.

Date	Downloads
Dec-06	75
Jan-07	104
Feb-07	115

Data Source: [81]

About the author

Cameron Shorter is on the Project Management Committee of the Mapbuilder project

---

MapGuide Project Update

by MapGuide Project Steering Committee

About MapGuide

MapGuide Open Source is an advanced platform for web-based geospatial application development, data update, and publishing. MapGuide incredibly flexible, being cross-platform (Windows and Linux) and cross-language (PHP, Java, and ASP.Net). It includes a strong geospatial processing API, and a large number of output formats, such as image (PNG, JPG, etc), WMS, WFS, and KML. In addition, by leveraging the flexible FDO Open Source data abstraction library, MapGuide can access multiple data sources, such as MySQL, PostGIS, Oracle, ArcSDE, Shape Files, SDF, WMS, WFS, and many image formats.

Recent Events

Since its initial release, there have been several significant events

- November 2005
  - Autodesk announces that new MapGuide code will be released as open source.
  - Technology preview release of MapGuide Open Source (under the ill-fated name MapServer Enterprise)
- February 2006
  - MapGuide code is made available through the Open Source Geospatial Foundation
  - MapGuide Open Source 1.0.0 is released
  - MapGuide development team starts using public issue tracking and feedback mechanisms
- July 2006
  - MapGuide Open Source 1.0.1 is released
- September 2006
  - Provenance review is completed
- October 2006
  - MapGuide Open Source 1.0.2 is released
  - MapGuide Open Source Project Steering Committee is formed. MapGuide Open Source is now controlled by a group of developers and users, the majority of which are not Autodesk employees.
  - First official RFC (Request For Change) is posted for review
- January 2007
  - MapGuide Open Source 1.1.0 is released
  - Open source GDAL Raster provider replaces proprietary Autodesk Raster Provider
  - A flexible development tracking system (Trac) is implemented. This allows users to more easily submit bug reports and enhancement requests.
- March 2007
  - MapGuide graduated from OSGeo Incubation, showing an acceptable level of participation, freedom from code encumbrances, and use of standard open source development methodologies.

[81]GML Viewer downloads: http://sourceforge.net/project/stats/detail.php?group_id=35246&ugn=mapbuilder&type=prdownload&mode=alltime&package_id=116388

Under Development

In the past months, several exciting new features have been added to MapGuide Open Source

- Native KML publishing, including support for regions, view-based scale ranges, and extrusion.
- Introduction of a highly configurable stylization engine, allowing for complex custom point and line styles

Statistics

- In the first three months of 2007, statistics show approximately
  - 2300 Windows downloads
  - 900 Linux downloads
  - 1200 PHP developer sample downloads
  - 1200 .Net developer sample downloads
  - 750 Java developer sample downloads
- mapguide-users mailing list
  - 470 members
  - averaging about 20 messages/day
- mapguide-internals (core development) mailing list
  - 130 members
  - averaging about 8 messages/day
- 17 RFCs (Requests for Change) submitted and adopted
- More useful stats can be found on the Ohloh site: `http://www.ohloh.net/projects/4656`

MapServer Project

by Daniel Morissette

About MapServer

MapServer is a development environment for building spatially-enabled web mapping applications and services. It is fast, flexible, reliable and can be integrated into just about any GIS environment. It supports many popular spatial data formats including OGC web services.

MapServer features MapScript, a powerful scripting environment, that supports PHP, Python, Perl, C#, Java and more. Using MapScript makes it fast and easy to build complex geospatial web applications.

Recent Events

October 2006

- MapServer 4.10.0 release

February 2007

- MapServer 4.10.1 release

Future development - MapServer 5.0

The MapServer development team plans to release version 5.0 by the end of summer 2007. The 5.0 Release Plan outlines a number of planned updates, including:

- Support for AGG, an up and coming image renderer as an alternative to GD
- Upgrade of WMS support to version 1.3.0
- Improved OGC SOS Server support
- New OGC OWS Common support version 1.0.0
- Dynamic charting capability
- Label prioritization
- MapScript memory management fixes
- Redesign of the LOG/DEBUG output mechanism
- ... and much more ...

Statistics

Mapserver-Users list statistics (for 2007-Q1):

- Number of subscribers: 2344
- Number of countries represented: 67 (In all continents but Antarctica)
- Number of posts per month: 415 (average for Jan-Mar 2007)

Also take a look at MapServer Development statistics at Ohloh. Here are some excerpts:

- **27 developers:** Over the past twelve months, 12 developers contributed new code to MapServer. This is a relatively large team, putting this project among the top 5% of all project teams on Ohloh. For this measurement, Ohloh considered only recent changes to the code. Over the entire history of the project, 27 developers have contributed.

- **Mature, well-established codebase:** The first lines of source code were added to MapServer in 2000. This is a relatively long time for an open source project to stay active, and can be a very good sign. A long source control history like this one shows that the project has enough merit to hold contributors's interest for a long time. It might indicate a mature and relatively bug-free code base, and can be a sign of an organized, dedicated development team.

Note: The source code for MapServer is actually older than the source control history. The project started around 1996 but started using CVS only in 2000.

Quantum GIS Project Update

Gary Sherman

Current and Upcoming Release

The QGIS development team released version 0.8.0 on December 29th 2006 after a long development period. Much of the time was spent in porting the code base from Qt 3.3 to Qt 4.2.x. With that effort complete, the project is staged to release new versions in a more timely fashion. Version 0.8.1 is scheduled to be released by the end of the first quarter of 2007. While primarily a bug fix release, there are will be a few minor enhancements, including an expanded set of tools in the GRASS toolbox.

Change in the Build System

The QGIS project has migrated to using CMake rather than the GNU autotools. This will become official with the 0.9 release, but the changes have been backported to the 0.8.1 branch as well. CMake has proven to be easy to use, flexible, and actually results in faster compilation times of the QGIS code base.

What's Ahead in Version 0.9

At version 0.9, Python bindings are included for most classes in the QGIS API. This means you can write both plugins and standalone programs using Python and PyQt. A number of folks are already developing plugins and the famous "Tim Sutton C++ tutorials" have been ported to Python as well by Martin Dobias. Check out the QGIS blog [82] for the tutorials and more information.

There are a number of substantial changes in 0.9, including a refactoring effort to support the use of the QGIS libraries in your custom programming projects. The development team is still working on the details of the 0.9 release. Keep an eye out for more details on the website as the work proceeds.

If you are adventurous, you can try out the new features and the Python bindings by checking out the development (that means unstable) code using svn [83]

QGIS Enters OSGeo

QGIS applied for and was accepted into incubation as an OSGeo project in late February. A poll of the community indicated that most people were in favor of membership. We now start the process of working with the OSGeo mentor to become a full fledged OSGeo project.

[82]qgis blog: `http://blog.qgis.org`
[83]qgis svn: `https://svn.qgis.org/repos/qgis/trunk/qgis`

2007 FREE AND OPEN SOURCE SOFTWARE
FOR GEOSPATIAL (FOSS4G) CONFERENCE
VICTORIA CANADA SEPTEMBER 24 TO 27, 2007

FOSS4G - Conference Registration Open

We are pleased to announce that online registration for the 2007 Free and Open Source Software for Geospatial conference (FOSS4G 2007) is now open. FOSS4G 2007 is the annual event bringing together the people and companies who create, use, and support open source geospatial software. Register now online.[84]

Register prior to the July 27th Early Bird Deadline to save on your registration fees! Take advantage of the opportunity FOSS4G 2007 offers to network with fellow geospatial data professionals, renew old acquaintances and make new ones.

For up-to-date information, registration and/or to submit a presentation, please visit the conference website.[85]

EXHIBITOR & SPONSORSHIP OPPORTUNITIES

For information on exhibitor and sponsorship opportunities, see the sponsors page[86] or contact Paul Ramsey, Conference Chair via email.[87]

SUBMIT A PRESENTATION

You can submit a presentation online.[88] The deadline for submissions is June 29th 2007.

FOSS4G presentations are 25 minute talks, with 5 minute question and answer sessions at the end. Presentations cover the use or development of open source geospatial software. Anyone can submit a presentation proposal and take part in the conference as a presenter. More information is available on the presentations page on the website.

We hope to see you in Victoria, Canada in September!

[84] Register online at: http://www.foss4g2007.org/register/
[85] Conference website: http://www.foss4g2007.org/
[86] Sponsors page: http://foss4g2007.org/sponsors
[87] Email Paul Ramsey at: pramsey@foss4g2007.org
[88] Submit a presentation at http://www.foss4g2007.org/presentations/

Editor in Chief:
Tyler Mitchell - tmitchell AT osgeo.org

Editor, News:
Jason Fournier

Editor, Case Studies:
Micha Silver

Editor, Project Spotlights:
Martin Wegmann

Editor, Integration Studies:
Martin Wegmann

Editor, Programming Tutorials:
Landon Blake

Acknowledgements
Various reviewers & the GRASS News Project

The *OSGeo Journal* is a publication of the *OSGeo Foundation*. The base of this newsletter, the LaTeX 2ε style source has been kindly provided by the GRASS and R News editorial board. All articles are copyrighted by the respective authors. Please use the OSGeo Journal url for submitting articles, more details concerning submission instructions can be found on the OSGeo homepage.

Newsletter online: http://www.osgeo.org/journal

OSGeo Homepage: http://www.osgeo.org

Mail contact through OSGeo, PO Box 4844, Williams Lake, British Columbia, Canada, V2G 2V8

ISSN 1994-1897