OSGeo Journal

The Journal of the Open Source Geospatial Foundation Volume 3 / December 2007

Proceedings of FOSS4G 2007

Integration & Development

- Portable GIS: GIS on a USB Stick
- Automatic Generation of Web-Based GIS/Database Applications
- db4o2D — Object Database Extension for 2D Geospatial Types
- Google Summer of Code for Geoinformatics

Topical Interest

- A Generic Approach to Manage Metadata Standards
- Towards Web Services Dedicated to Thematic Mapping
- Interoperability for 3D Geodata: Experiences with CityGML & OGC Web Services
- A Model-Driven Web Feature Service for Enhanced Semantic Interoperability
- Spatial-Yap: A Spatio-Deductive Database System

Case Studies

- DIVERT: Development of Inter-Vehicular Reliable Telematics
- GRASS GIS and Modeling of Natural Hazards: An Integrated Approach for Debris Flow Simulation
- A Spatial Database to Integrate the Information of the Rondonia Natural Resource Management Project
- GeoSIPAM: Free & Open Source Software Applied to the Protection of Brazilian Amazon
- The Amazon Deforestation Monitoring System: A Large Environmental Database Developed on TerraLib and PostgreSQL

From the Editor...

As promised, here it is — the conference proceedings from the biggest and best FOSS4G conference yet. This is a special edition of the OSGeo Journal, dedicated to bringing you a very small sampling of some of the hundreds of papers and topics presented at the event. For more information, conference summaries and discussion see the various blogs and reports available on the reviews page of the `foss4g2007.org` web-site.

The local organising committee is already well into preparing for next year's event in Cape Town, South Africa. For more information see the `foss4g2008.org` web-site. Meanwhile, as you read this, the contenders for the FOSS4G **2009** bid are speedily writing up their proposals for hosting the event — best wishes to all proposals!

The next issue of the Journal will be back to normal. This is your invitation to contribute an article, code examples, tutorial, case studies and more. If you are interested in submitting an article, please add yourself to the Volume 4 wiki page[1] and an editor will be in touch with you. Contact me directly if you are wondering how your article might fit into the Journal.

As always, thank you to the rest of the Editorial team, proof-readers and **the 37 contributors** to this volume. Enjoy the articles!

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

Contents of this volume:

From the Editor... 1
FOSS4G 2007 Sponsors & Supporters 2
Portable GIS: GIS on a USB Stick 3
Automatic Generation of Web-Based GIS/-Database Applications 5
db4o2D - Object Database Extension for 2D Geospatial Types 17
Google Summer of Code for Geoinformatics . 21
A Generic Approach to Manage Metadata Standards 24
Towards Web Services Dedicated to Thematic Mapping 31
Interoperability for 3D Geodata 34
A Model-Driven Web Feature Service for Enhanced Semantic Interoperability 38
Spatial-Yap: A Spatio-Deductive Database System 44
The DIVERT Project: Development of Inter-Vehicular Reliable Telematics 48
GRASS GIS and Modelling of Natural Hazards 53
A Spatial Database to Integrate Information of the Rondonia Natural Resource Management Project 60
GeoSIPAM 64
The Amazon Deforestation Monitoring System 70

[1] `http://wiki.osgeo.org/index.php/Journal_Volume_4`

FOSS4G 2007 Sponsors & Supporters

Reflections from two supporters

The Free and Open Source Software for Geospatial Conference (FOSS4G), with the involvement of the Open Source Geospatial (OSGeo) Foundation, has become the premiere event for the open source geospatial community. This year I was asked to provide a "wrap up" as part of the closing session. These are the themes I presented, which all neatly fall under an overarching theme of "maturity."

The experiences of the last few years (from the development of projects, to the creation of OSGeo, among other things) have given the FOSS4G community a confidence that I didn't feel in the past. While I've missed a few meetings in this series, notably the last edition of this event, the 2007 event struck me as a "real" conference. I see a change in the answer to this question, which I've posed to attendees over the last few years: "What software stack are you working with?" I suggest that the proportion of attendees new to GIS or new to geospatial was rather high. The FOSS4G community is "just like" the rest of the GIS community. I would identify the pursuit of effective business models and development funding as far "calmer" than in past years. The demand for open source geospatial education is growing.

Adena Schutzberg
Executive Editor, Directions Magazine

The Open Planning Project wishes to thank FOSS4G 2007 conference organizers and especially attendees for making a terrific event. We had a great time connecting with colleagues and meeting new users of our collective technologies.

We were particularly excited to see the increasing popularity of GeoServer and OpenLayers, and enjoyed the opportunity to demonstrate the new feature development that we have been supporting in those projects. We look forward to another great year of advances in the open geo-stack and will be back with more exciting developments at the next conference.

Chris Holmes, The Open Planning Project

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

Thank you to all the FOSS4G 2007 Sponsors for their support!

Autodesk
Google Refractions Research
British Columbia Integrated Land Management Bureau
Safe Software
DM Solutions
Leica Geosystems
ESRI
City of Nanaimo
Orkney
OGC
Sierra Systems
GeoConnections GéoConnexions

Integration & Development

Portable GIS: GIS on a USB Stick

Jo Cook

Summary

This is a suite of GIS programmes that can be run from a USB stick (and hence are totally portable) amongst computers running Microsoft Windows. This leads to a package that can be quickly and easily deployed by individuals with no prior experience; in a range of different environments; and provides an easy and cost-efficient method for companies and individuals to evaluate these packages in comparison to costly proprietary alternatives.

Introduction

The term "Portable GIS" refers to a full suite of Geographic Information Systems (GIS) Programmes that can be run from a USB stick. This allows them to be used on any Windows PC with no need for installation or configuration. Some alterations were made to the existing code, where necessary, to allow the programmes to run in this way, such as removing the need for a fixed drive letter. Furthermore the individual packages have been worked into a single "suite", with a menu system for operating them, and full documentation.

Open source programmes are often difficult to install compared to proprietary Windows alternatives. Developers often seem to assume a much greater level of prior knowledge than the average new user has, and options for getting help, such as mailing lists and online discussion groups can be intimidating. I believe, from personal experience and anecdotal evidence, that many new users are put off using open source software because they cannot install it, let alone use it.

Live CDs are often used in these situations, but the advantage to the "Portable GIS" approach is that it works in the same operating system that the user is familiar with, with access to their file system (and hence data) rather than in a self-contained sandbox environment.

There are two further advantages to the "Portable GIS" approach. The fact that it runs on a USB stick means that it can be deployed quickly and easily in a number of situations, and the fact that it requires no

end-user installation and configuration makes it an attractive proposition for companies and individuals wishing to evaluate open source GIS.

Products Included on the stick

The programmes can be split into two modules, desk-based and web-based. Every aspect of the suite is open source, including the documentation and the menu system. The programmes are as follows:

Desk-based: GRASS, QGIS, FWTools
Web-based: XAMPPlite (full apache/MySQL/PHP web server stack), PostgreSQL with PostGIS, MapServer, OpenLayers, Tilecache.

The documentation has been written in the form of a tiddlywiki. This is single-page html wiki, so can be edited by users to add their own notes as they see fit. The menu system is based on the "daily cup of tech" menu, built by Tim Fehlman. It is built using autoit, and runs from the system tray. It provides links to the setup programmes (where necessary), and to the start and stop executables for each package. It also allows the user to browse the drive file system in the normal fashion.

Problems and Learning Curves

Whilst some of the programmes, such as XAMPPlite, were designed to run in this way, and others, such as QGIS were known to work, others needed a degree of alteration in order to run without the need for a fixed drive letter. In many cases this was achieved by editing the batch files that set up the working environment for the programme, or by installing in the traditional way on a Windows PC and copying the programme folder over (the trial and error approach).

Furthermore, the native Windows installation of GRASS is complex. Understanding the syntax of batch files, and in particular how programmes like GRASS use them, was the main problem that had to be overcome in the construction of the "Portable GIS" suite.

In other cases, such as PostgreSQL, where traditionally a particular user was required to run the programme, the situation appeared hopeless until a chance post on the mailing list indicated that this was no longer the case. PostgreSQL could now be installed and run as the user logged on to the PC, and no longer needed to be run as a service. This made the process of installation on a USB stick relatively simple.

Some problems still remain. Windows machines in work or academic environments are often locked down to prevent services from running or setting environment variables. In these cases, many aspects of the suite will not work. Currently it is difficult to see a way round this issue, other than to bring it to the attention of end users!

The way forward

As well as generally streamlining the suite, there are several ways forward. More programmes can be added, where there is a need. One issue with this is the increasing size of the package (currently 800MB).

A transparent and straightforward procedure for updating the individual packages needs to be identified. Either detailed documentation for updating each package needs to be provided, including those files that should NOT be updated (those that have been modified to allow the programme to run portably for example), or the author needs to take responsibility for producing a new suite when programmes are updated.

It may be possible to streamline the installation by removing duplicate libraries and hence reduce the size of the overall package. For example, mapserver is included both as a web programme and also in the FWTools application, and GRASS is included in it's own right and as part of the QGIS installation. However, this would prevent users from updating individual programmes as outlined above.

Requests have been made to provide cross-platform versions of the suite. This would be an interesting challenge, but to a certain extent it is assumed that Linux users will be more familiar with the installation of open source software and the community around it!

At the moment it is not publicly available due to it's large size and the bandwidth required for hosting. Obviously if it is to be used, then it needs to be hosted. Various options for this are available and being investigated at this time. For more information see the Archaeogeek blog discussion thread.[2].

Jo Cook
Oxford Archaeology North
http://www.oxfordarchaeology.co.uk
j.cook AT oxfordarch.co.uk

[2]Archaeogeek blog discussion thread: http://www.archaeogeek.com/blog/portable-gis/

Automatic Generation of Web-Based GIS/Database Applications

Nirut Chalainanont, Junya Sano and Toshimi Minoura

Abstract

We have been developing *web-based GIS/database* (WebGD) applications that allow users to *insert*, *query*, *update*, and *delete geographical features* and the data associated with them from standard Web browsers. The code shared by these applications is organized as the WebGD framework. The behavior of the map interface of a WebGD application is defined by *configuration files*. We have built also the WebGD application generator (WebGD-Gen) that *automatically* produces those configuration files from the database metadata, including those in tables `geometry_columns` and `spatial_ref_sys`. WebGD-Gen can generate also Web scripts that interact with the map interface and the database. Thus the WebGD framework and WebGD-Gen can significantly reduce the development time and the maintenance cost of a complex Web-based GIS/database application.

Introduction

The Internet has become the major venue for sharing information. A dynamic Web-based mapping application allows users without desktop-based GIS software to perform *spatial queries* and produce maps with standard Web browsers.

However, a typical web-based GIS application allows only the retrieval of maps and map-related data. A web server provides information to the client, but the client cannot feed information back to the server (3). This *unidirectional* flow of information is a major problem with a current typical map-server application. Furthermore, creating an interactive web application with a map interface is time-consuming. Commercial map servers and geographical database management systems are expensive. This situation hinders use of Web-based GIS applications in data gathering, analysis and decision-making.

We have been developing a set of tools that significantly reduces the cost of application development (14; 16; 9; 15). The code shared by interactive Internet GIS applications is organized as the Web-based GIS/database (WebGD) framework. Most of the complex workings for delivering GIS functions over the Web are included in this framework. With the WebGD framework, we can create a map interface through which users can *insert*, *query*, and *delete* geographical features and the data associated with them only by providing *configuration files*.

An important feature of a WebGD application is that it tightly integrates spatial data and associated tabular data, enabling analyses involving location-based data (13). These functions are available to users at different geographical locations for economical and timely data management.

We developed several years ago a *Web-form generator* that automatically generates scripts for traditional Web-forms from the *schema* of a relational database (4). This functionality was recently extended as the WebGD application generator (WebGD-Gen). The new application generator can produce most of the code for an entire WebGD application from the schema of a relational database and the information on *geometry columns*. With the map interface and the Web scripts automatically generated, geographical features (i.e., points, linestrings and polygons) can be *inserted*, *queried*, and *deleted*. Thus, when a relational schema and the GIS metadata for the map layers are available, a *non-customized* application can be quickly assembled with the WebGD framework and the WebGD-Gen application generator.

The automatic generation of the map interface and Web-form scripts make possible *incremental* and *iterative* development of complex web-based GIS applications. When a map layer is added or modified, we only need to create or update the configuration file for that layer and then regenerate the Web forms for that layer. When a database schema is modified, we can regenerate the entire set of Web scripts in several minutes. Therefore, even with an incomplete set of map layers and a database schema, we can generate a working prototype for an initial review. Furthermore, we can fix most of the software bugs by modifying only the shared code in the WebGD framework or WebGD-Gen and then by regenerating Web scripts for each application.

WebGD applications use the following *open-source* software components. PostgreSQL, an *object-relational* database, and PostGIS together manage geospatial data. PostGIS is an extension of Post-

greSQL for GIS applications (11). MapServer (12) generates maps to be displayed on a web browser by using geospatial data provided by PostGIS. Web pages, including the one that displays the maps, are generated by server-side scripts written in PHP. The PHP MapScript module interacts with MapServer (6; 2). When a request to insert or delete a map feature is received by a PHP script, the script directly accesses the PostgreSQL database, using the PostGIS extension. An application developed runs on a PC without any licensed software.

We explain the features supported by the WebGD framework in Section **WebGD Applications**. The process of generating map-layer configuration files is explained in Section **WebGD Framework**, and that of generating Web scripts in Section 4. The process of automatic generation of map-layer configuration files is explained in Section **Automatic Generation of Map-Layer Configuration Files**. In Section **WebGD Development History**, we describe a brief history of the development of the WebGD framework and WebGD-Gen. Section **Conclusions and Future-Work** concludes this paper.

WebGD Applications

The Web interface of one of the WebGD applications, Natural Heritage Information System (NHIS) for North Carolina, is shown in Figure 1. This application provides a map interface for a copy of the Biotics 4.0 database maintained by the North Carolina Natural Heritage Program. Biotics 4.0 is a desktop GIS application built on the database developed by NatureServe. The key elements in this database are *element occurrences* (EOs), which are *areas* of land and/or water in which species are, or were present (7). EO records have both *spatial* and *tabular* data, and the database contain approximately 700 relational tables (5). The Biotics Mapper implemented with ArcView by NatureServe provides a map interface that allows EO representations and associated data to be created, updated, and deleted (8). In our implementation, we can perform these operations with standard Web browsers. Also, Web forms, approximately 3500 in total, are provided for all tables in the database.

The NHIS application enables *bi-directional* movement of *geospatial data* as well as ordinary data. Scientists and others with proper authentication can *insert*, *query*, and *delete* geographical features such as EO polygons, lines, and points, as well as the data associated with them. Queries can be executed by spatially *selecting an area* on the map or by using a traditional web form. In addition, one-meter resolution *digital orthographic quadrangles* DOQ, or aerial images, are included as a layer. When DOQ images are combined with other map layers such as highways, county boundaries, streams, and streets, locations can be easily pinpointed by taking advantage of features between map layers (16).

The major operations supported by the map interface of a WebGD application are as follows:

1. To retrieve information on the geographical features in the area of interest, the user can zoom in/out to that area by using the map navigation tools. If the user zoom-in enough, one-meter resolution aerial photos are displayed. The user can also go to a new area by selecting an entry in the **Quick View** menu.
2. To get information about a geographical feature, the user can select a layer in the legend and **Information** in the function menu, and then click the boundary of the feature.
3. Function **Insert** allows a geographical feature to be added with mouse clicks on the map. **Done** need be pressed after all points are entered.
4. Function **Search by Area** allows the user to retrieve the list of features that are within a *bounding box* specified on the map and that satisfy a search condition. The features that satisfy the search condition are *highlighted* on the map. Furthermore, the user can select features in the list by marking the checkboxes associated with them. Then, if the map is refreshed, the selected features are highlighted.
5. The data administration interface can be activated by clicking on the **Database** entry in the menu bar below the banner. A tree icon can be clicked to display a *treeview* for browsing. The treeview for **Higher Taxonomy** is the major one. To access the data of this application, a user must login with a password as some data on endangered species are confidential.

Several WebGD applications created with the WebGD framework and WebGD-Gen can be accessed from this page.[3]

[3]WebGD sample applications: `http://yukon.een.orst.edu/index_webgd.html` You may insert/query/update/delete data. However, please DO NOT disturb existing data. If login is required, use user name `cs540` and password `CSxyz540`. You may have to login twice, once for the server and then for an application.

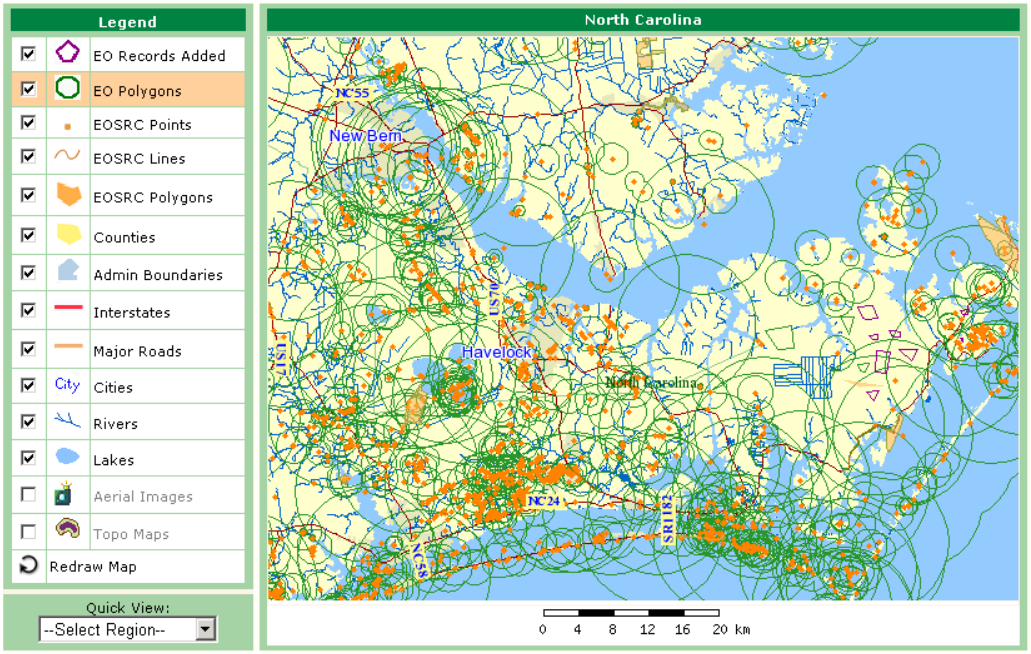


Figure 1: Interface of the NHIS Application for North Carolina

The first application is Natural Heritage Information System. Although this application can cover the whole USA or the world, the data are currently available only for North Carolina. The second application provides a map interface for an application that keeps track of conservation practices on land parcels. The third one is a Web-based mapping application for plant germplasm collection maintained at Western Regional Plant Introduction Station (USDA-ARS). The fourth one allows the soil information at the location where a mouse click occurs on the map interface to be retrieved.

One salient feature of the current WebGD framework is *dynamic switching of spatial references*. Typically, different geographic regions and localities have preferred *map projections* in order to avoid distortions in the maps created (1). The framework allows the whole world to be covered with multiple-levels of maps, e.g., the world map, continent maps, and region maps. The map interface then automatically selects the most suitable projection for the region whose portion is displayed. For example, the world can use the *geographical coordinate system*, the United States the *Albers equal-area projection*, and Oregon the *Lambert conformal conic projection*. Thus, spatial analysis can be performed with the most appropriate projection for a particular area. The *dynamic switching* of the *spatial reference*, the *map file*, the *legend*, and the *quick view menu* supported by the current WebGD framework allows any part of the world to be covered with its own scale and spatial reference, including regions with one-meter resolution aerial images. This is a very important feature, especially now that the cost of storing aerial images for the entire US has dropped to affordable levels (10 terabytes needed to store aerial images for the entire US now cost around $10,000). Furthermore, many states are putting aerial images in the public domain.

WebGD Framework

The WebGD framework supports common features required by the map interface of a WebGD application such as zoom in/out, pan, and insert/query/update/delete operations of geographical features. The organization of a WebGD application is shown in Figure 2.

A map operation that does not manipulate geometry features, e.g., *zoom-in*, *zoom-out*, or *panning*, is processed as follows:

1. The user action on the map is transmitted from the Web browser to the Web server as an HTTP GET request.
2. The Web server activates a PHP script that handles the user action.
3. Inside the PHP script, various methods in PHP MapScript are called to prepare the new map

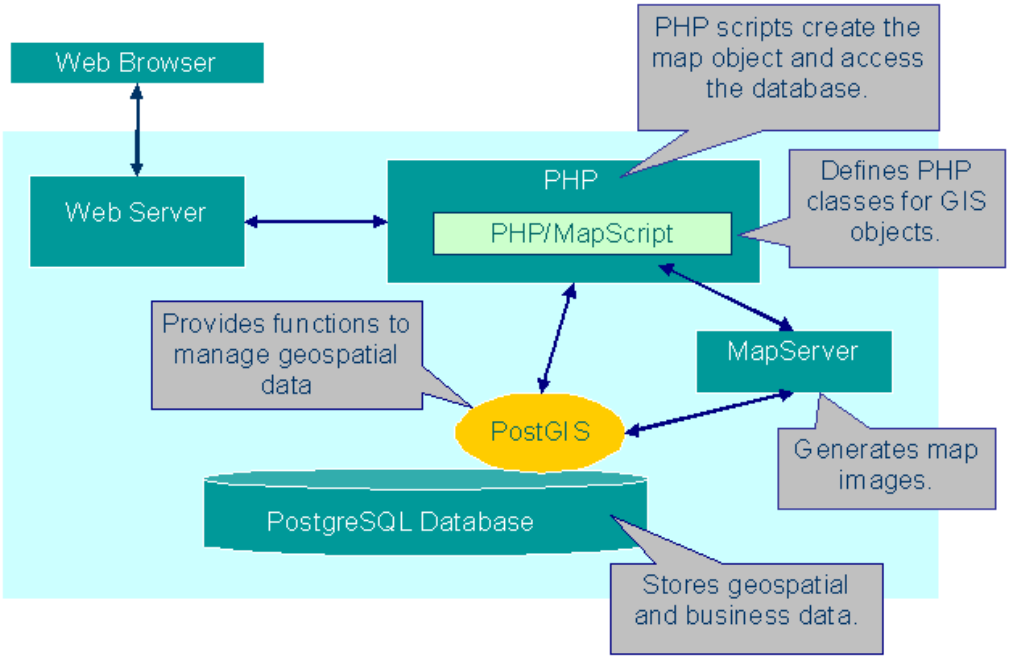


Figure 2: Organization of a WebGD Application

parameters, such as the special reference and the extent of the map and the names of the layers to be displayed. The map drawing method in PHP MapScript is then called to instruct the MapServer to create the map image.

4. The MapServer requests data for the map layers whose data are stored in the PostgreSQL database.
5. The data for the layers are returned.
6. The map image created is returned to the PHP script.
7. The HTML page generated by the PHP script is returned to the Web server.
8. The Web server transmits the HTML page including the new map image to the Web browser.

For a map operation that accesses or manipulates geometry features, i.e., *search by area*, *insert*, or *move point*, is processed as explained above, except between steps 2 and 3 the following operations are performed.

a The PHP script connects to the PostgreSQL database to perform a spatial operation with PostGIS-enabled SQL statements.
b The result of the spatial operation is returned to the PHP script by the SQL statements.

Furthermore, *configuration files* are user extensively as part of the WebGD framework for application customization. Major configuration files are *region configuration files*, *map layer configuration files*, and a *quick view configuration file*.

Region Configuration Files

In order to minimize map distortion, different map areas can be displayed with different spatial references. The *current region* is determined to be the *smallest* one that encompass the extent of the map to be generated. The *spatial reference* is then automatically switched to that of the new region. Then the layers in the map legend, the list of the map-navigation and data-manipulation commands, and the *quick view* list are reconfigured for the new region. These reconfigurations are necessary because different regions may require different sets of map layers, command, and quick view list.

A *region configuration file* for each map region includes the following definitions:

1. the special reference for the region,
2. the *name of the region* displayed on the map interface,
3. the *unit of distance measurement*,
4. the *name of the map layer configuration file* for the region, and
5. the *name of the quick view configuration file*.

For example, the region configuration file for the world is as follows:

```
$region = array(
   "gid" => 1,
   "name" => "world",
   "display_name => "World",
   "srid" => 4326,
   "rank" => 1,
   "units" => "MS_DD",
   "mapfile" => "gmap75_world.map",
   "quickview" => "world_gview.php",
   "legend" => "world_maplayers.php",
   "proj4text" => "+proj=longlat +ellps=WGS84
         +datum=WGS84"
);
```

The following options can be specified in a configuration file:

gid – the unique number assigned to each region, which is the primary key value of the row representing the region in table `regions` in the database.

name – the name of the region for programming.

display_name – the name of the region used on the map interface. This name may include capital letters, white spaces, and other punctuation marks.

srid – the *spatial reference identifier* for the map projection assigned with this region.

rank – the priority number used in determining the region for the map area to be viewed. The region with the highest rank number is selected among the regions that completely encompass the map area to be viewed. Regions with higher rank numbers cover smaller areas.

units – the unit of measurement associated with the region.

mapfile – the map file to be loaded when this region is selected.

quickview – the name of the quick view configuration file for the region.

legend – the name of the map layer configuration file for the region.

proj4text – the projection string used for the region.

Whenever the map region changes due to a user action on the map interface, the region configuration file for the new region is loaded dynamically. Then the map interface is customized according to the new region configuration file.

The region configuration file for Oregon, for example, contains the following definitions:

```
$region = array(
   "gid" => 150137,
   "name" => "oregon",
   "display_name" => "Oregon",
   "srid" => 6010,
   "rank" => 150137,
   "units" => "MS_FEET",
   "mapfile" => "gmap75_oregon.map",
   "quickview" => "oregon_gview.php",
   "legend" => "oregon_maplayers.php",
   "proj4text" => "+proj=lcc +lat_1=43.0
         +lat_2=45.5 +lat_0=41.75
         +lon_0=-120.5
         +x_0=400000.00000 +y_0=0.0"
);
```

Map Layer Configuration Files

The *map layer configuration file* provided for a region specifies the layers to be included in the legend and their characteristics. It is possible for multiple map regions to share one map layer configuration file.

The map layer configuration file for the Oregon region, for example, contains the following definitions:

```
$layer_groups = array (
  'grp_eo_py' => array(
     'geom_type' => 'polygon',
     'table' => 'eo_py',
     'layer_selectable' => true,
     'gid_column' => 'gid',
     'geom_col' => 'the_geom',
     'legend_label' => 'EO Polygons',
     'search_script' =>
      'forms/eo/eo_py_eo_search.phtml',
     'select_script' =>
      'forms/eo/eo_py_eo_select.phtml',
     'edit_script' =>
      'forms/eo/eo_py_edit.phtml',
     'normal_layer' => 'eo_py',
     'searched_layer' => 'eo_py_searched',
     'checked_layer' => 'eo_py_checked',
     'selected_layer' => 'eo_py_selected',
     'img_src' => 'images/eo_poly.png',
     'img_width' => 26,
     'img_height' => 26,
     'onclick' => 'activate_layer("grp_eo_py")',
     'data_srid' => 32119
  ),
  'grp_eosrc_pt' => array(
     'geom_type' => 'point',
     'table' => 'eosrc_pt',
     'layer_selectable' => true,
     'gid_column' => 'gid',
     'geom_col' => 'the_geom',
     'legend_label' => 'EOSRC Points',
     'search_script' =>
      'forms/eo/eosrc_pt_search.phtml',
     'select_script' =>
      'forms/eo/eosrc_pt_select.phtml',
     'edit_script' =>
      'forms/eo/eosrc_pt_edit.phtml',
     'normal_layer' => 'eosrc_pt',
     'searched_layer' => 'eosrc_pt_searched',
     'checked_layer' => 'eosrc_pt_checked',
     'selected_layer' => 'eosrc_pt_selected',
```

```
    'img_width' => 5,
    'img_height' => 5,
    'onclick' => 'activate_layer("grp_eosrc_pt")',
    'data_srid' => 32119
  ),
...
```

According to the above configuration file, the map legend shown in Figure 3 is produced.

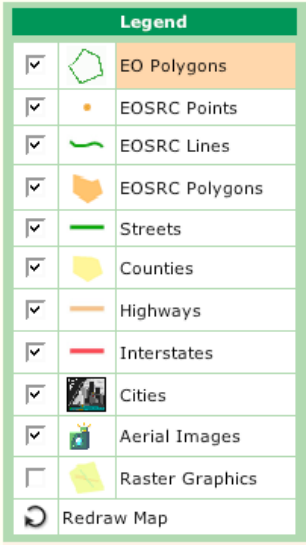


Figure 3: Map Legend of WebGD Application NHIS

We now explain each option used in map-layer configuration:

geom_type – the type of geometry features contained in the layer. The value can be `polygon`, `multipolygon`, `linestring`, `multilinestring`, `point` or `multipoint`. Exact spatial operations performed depend on this type. For example, if the type is `point`, a point is inserted on the map with an insert operation. On the other hand, if the type is `polygon`, a polygon feature can be inserted.

table – the name of the table in the database that contains the geometry column for the layer.

layer_selectable – the boolean option that determines whether spatial operations can be performed on the layer or not. Some layers, such as those for counties and highways in the above example, are static, and they are not selectable for spatial operations.

gid_column – the name of the column that contains the identifiers for the geometry features in the map layer.

geom_col – the name of the geometry column for the geometry features contained in the layer. The default name is `the_geom`.

legend_label – the name of the layer in the legend.

search_script – the name and the location of the search form associated with the layer.

select_script – the name and the location of the select form associated with the layer.

edit_script – the name and the location of the edit form associated with the layer.

normal_layer – the name of the map layer displayed when no highlighting occurs.

searched_layer, checked_layer, selected_layer – the names of the layers used to highlight the geometry features *searched for*, *checked*, or *selected*, respectively. The geometry features returned by a search operation is *searched for*, those whose checkboxes are turned on in search result form are *selected*, and the geometry feature chosen for editing is *selected*.

img_src, img_width, img_height – the file name, the width, and the height of the icon in the legend.

onclick – the name of the javascript event handler activated when the user selects the layer in the legend.

data_srid – the *spatial reference identifier (srid)* that designates the map projection used by the geometry features in the map layer. If this srid is different from that of the map region, then geometry features in the layer are reprojected before they are displayed on the map.

Quick View Configuration File

The *quick view* mechanism allows the user to select the map area of her interest. That is, the user can switch to a new map area quickly by selecting the map area from the quick-view dropdown list.

Each entry in a *quick view configuration file* describes the name of the map area represented by the entry, the map projection used by the *extent* of the map area, and the extent. The quick view configuration file for the Oregon region shown in Figure 4, for example, can be as follows:

```
$qview = array(
  array(
    'name' => 'World',
    'srid' => 4326,
    'extent' => '-180,-90,180,90'
  ),
  array(
    'name' => 'United States',
    'srid' => 4326,
    'extent' => '-125,13,-65,53'
  ),
  array(
    'name' => 'United States, East',
```

```
    'srid' => 4326,
    'extent' => '-102,22,-60,50'
  ),
  array(
    'name' => 'United States, West',
    'srid' => 4326,
    'extent' => '-135,30,-105,50'
  ),
  array(
    'name' => 'Whole Oregon',
    'srid' => 6010,
    'extent' => '46461.662375,-43912.968464,
      2487069.754184,1785176.331408',
  ),
...
```

The quick view list of regions as displayed in the map interface corresponding to the configuration file above:

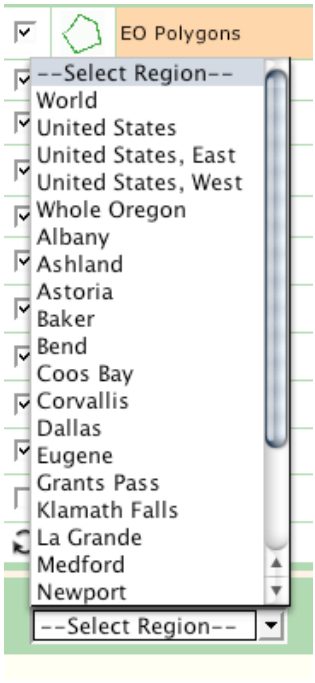


Figure 4: Quick View Selection of WebGD Application NHIS

Each entry in a quick view configuration file can specify the following options:

name – the name of the map area.
srid – the spatial reference identifier for the extent explained below.
extent – the `xmin`, `ymin`, `xmax`, and `ymax` values describing the positions of the lower left and upper right corners of the map area.

[4] 700 x 50

Automatic Generation of Web-Form Scripts

Several tools have been developed to augment the WebGD framework and simplify application development. The WebGD Web-site generator (WebGD-Gen) can create an entire WebGD application, including a web-based mapping interface. WebGD-Gen *automatically* generates a consistent set of Web scripts from *configuration files*, which are again automatically generated from a relational database schema. Since form generation is automatic, the cost of application development is greatly reduced. For a database such as Biotics that contains approximately 700 tables, programming all the required 3,500[4] forms manually can be very costly, even unfeasible.

WebGD-Gen is implemented as a collection of *templates*. Each template, combined with a corresponding *configuration file*, generates one of the following six types of Web scripts: *search*, *select*, *edit*, *information*, *action*, and *treeview* scripts. Templates and configuration files are written in PHP. The Web scripts generated by them are also in PHP. The generated scripts are executed on a Web server by a PHP interpreter. Each script, except for an action script, creates a Web form that is displayed on a client computer by a Web browser. Figure 5 illustrates the interactions among the Web scripts and forms.

Furthermore, WebGD-Gen can automatically generate the statements for inserting, searching, and deleting *geographical features* if the following lines, for example, are added to a configuration file:

```
 // type of geographical features
$web_gd = 'MULTIPOLYGON';
// layer group in legend
$layer_name = 'grp_eo_py';
// geometry column containing shapes
$geometry_column = 'the_geom';
//geographical feature IDs
$gid_column = 'gid';
// epsg spatial reference
$db_table_srid = 32119;
```

The forms generated for geographical features can perform the following additional functions compared to those for ordinary database tables:

1. A search form can be activated from a map interface. In this case, the extent of a search box specified on the map is passed as additional search parameters.
2. A select form includes additional JavaScript code for highlighting geographical features retrieved or selected by the user.

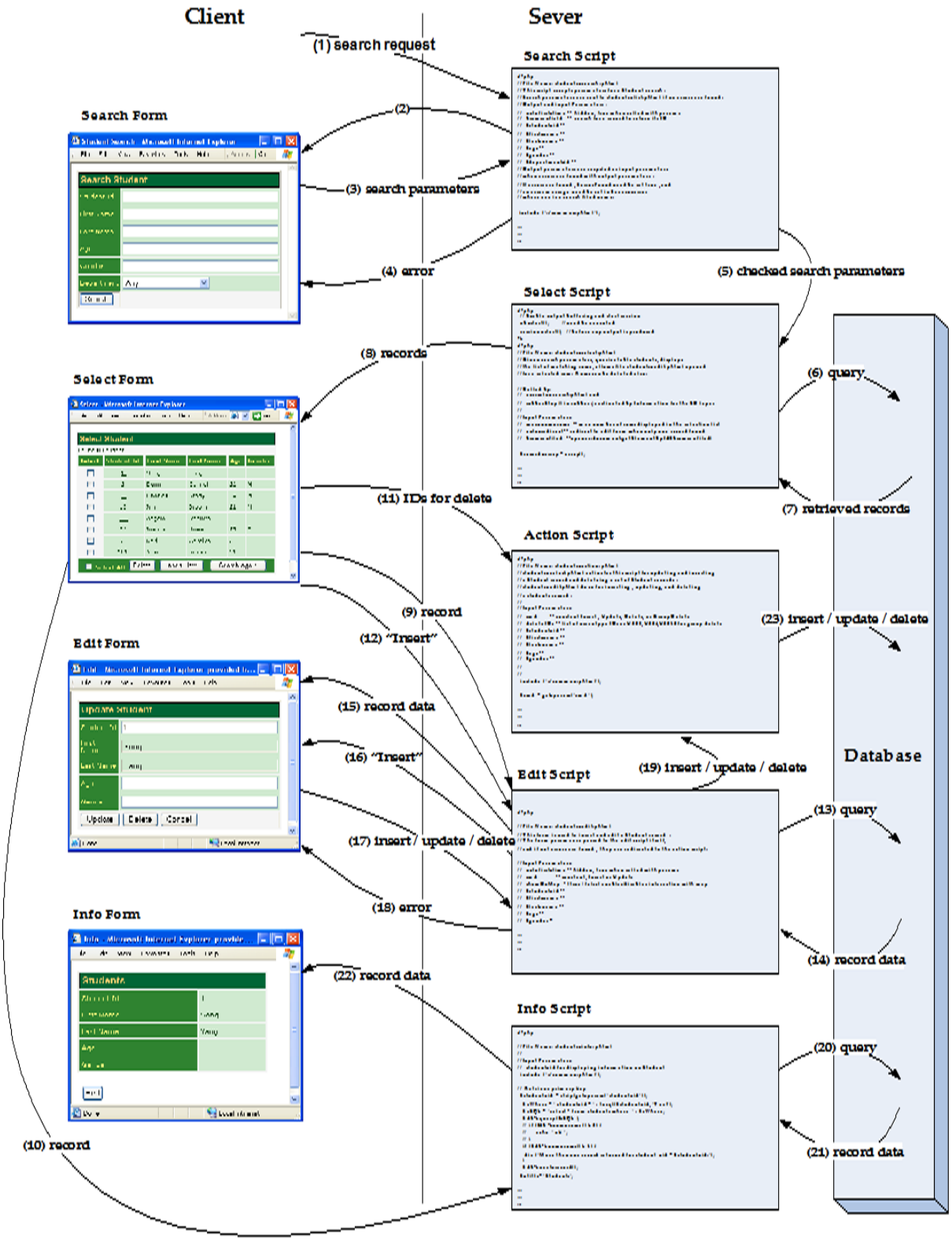


Figure 5: Interactions among Web Scripts and Forms

3. An edit form can insert a record for a geographical feature, after transforming the coordinate values from the spatial reference used by the current map interface to the one used by the geometry column for the record.

The forms related are automatically linked to each other. Figure 6 shows, as an example, the *edit* form for a Student table. From this edit form, the user can open the forms for the department and courses related to the student. The information needed to create the links are extracted from the *primary-key/foreign-key relationships* among the tables in the database.

In order to generate the Student edit form shown in Figure 6, the following entry is provided in the configuration file.

```
$edit_fields=array(
   array("column"=>"student_id",
     "label"=>"Student Id", "type"=>"numeric",
     "maxlen"=>"40", "size"=>"40"),
   array("column"=>"first_name",
     "label"=>"First Name", "type"=>"text",
     "maxlen"=>"40", "size"=>"40"),
   array("column"=>"age", "label"=>"Age",
     "type"=>"text", "maxlen"=>"40", "size"=>"40"),
   array("column"=>"gender", "label"=>"Gender",
     "type"=>"text", "maxlen"=>"40", "size"=>"40"),
   array("column"=>"department_id",
     "label"=>Department ID", "type"=>"to_one",
     "linked_table"=>"departments", "maxlen"=>"40",
     "size"=>"40"),
   array("label"=>"Courses Taken", "type"=>"to_many",
     "linked_table"=>"courses", "maxlen"=>"40",
     "size"=>"40"),
   array("column"=>"other_info",
     "label"=>"Other Information",
     "type"=>"textarea", "rows"=>"4",
     "cols"=>"32",
);
```

Each element in array `$edit_fields` represents a field in an edit form, with options `column`, `label`, `table`, `maxlen`, `size`, and `type`. Option `type` can be `numeric`, `text`, `time`, `date`, `email`, `phone`, `textarea`, `to_one`, or `to_many`.

textarea: The input is a string displayed in a text area.

to_one: A field of type `to_one` allows a user to view or modify the record related to the current record via a *one-to-one* or *many-to-one* relationship type. In our example, column `department_id` in table `students` is specified as `to_one`. When the `View` button is clicked, the edit form showing the record of the student's department is displayed. Furthermore, a user can search and select a new department record to be linked. Option `linked_table` designates the name of the table storing the associated record. Option `child_column` indicates the column for linking in the associated table. If this option is omitted, the name of the column for linking is identical to that of the foreign key column in the current record.

to_many: The purpose of this type is to list the records related to the current record via a *one-to-many* or *many-to-many* relationship type. In our example, the field labelled `Courses Taken` is specified to be `to_many`. When the `Show` button is clicked, all the courses taken by the student are displayed in the select form for courses. This type needs options `linked_table`, `parent_column`, and `child_column`. Option `linked_table` and option `child_column` can be omitted, when the name of the foreign key column of the associated table is identical to that of the primary key column of the current record. Option `parent_column` indicates the *foreign key* column in the current table. If this option is omitted, the foreign key column is the primary key column of the current table.

Automatic Generation of Map-Layer Configuration Files

We can semi-automatically create a map-layer configuration file by using information in the table `geometry_columns` that stores GIS-related metadata and the layer names in a map file. Figure 7 shows this process for region *xxx*.

1. The default *map-layer meta configuration file* *xxx*`_maplayers.mconfig`, which lists all the layer groups, can be created by **map_mconfgen**. The map-layer group names are generated from the table names stored in table `geometry_columns` and the map-layer group names defined in the map file.
2. We can customize the default map-layer meta configuration file, whose format is identical to a map-layer configuration file, by selecting and reordering the layer groups. We can also provide customized values for the parameters in each layer group. The meta configuration file needs to keep only the definitions of customized parameters.
3. Finally, map-layer configuration file *xxx*`_maplayers.config` is generated by **map_confgen**. The information required by this step is

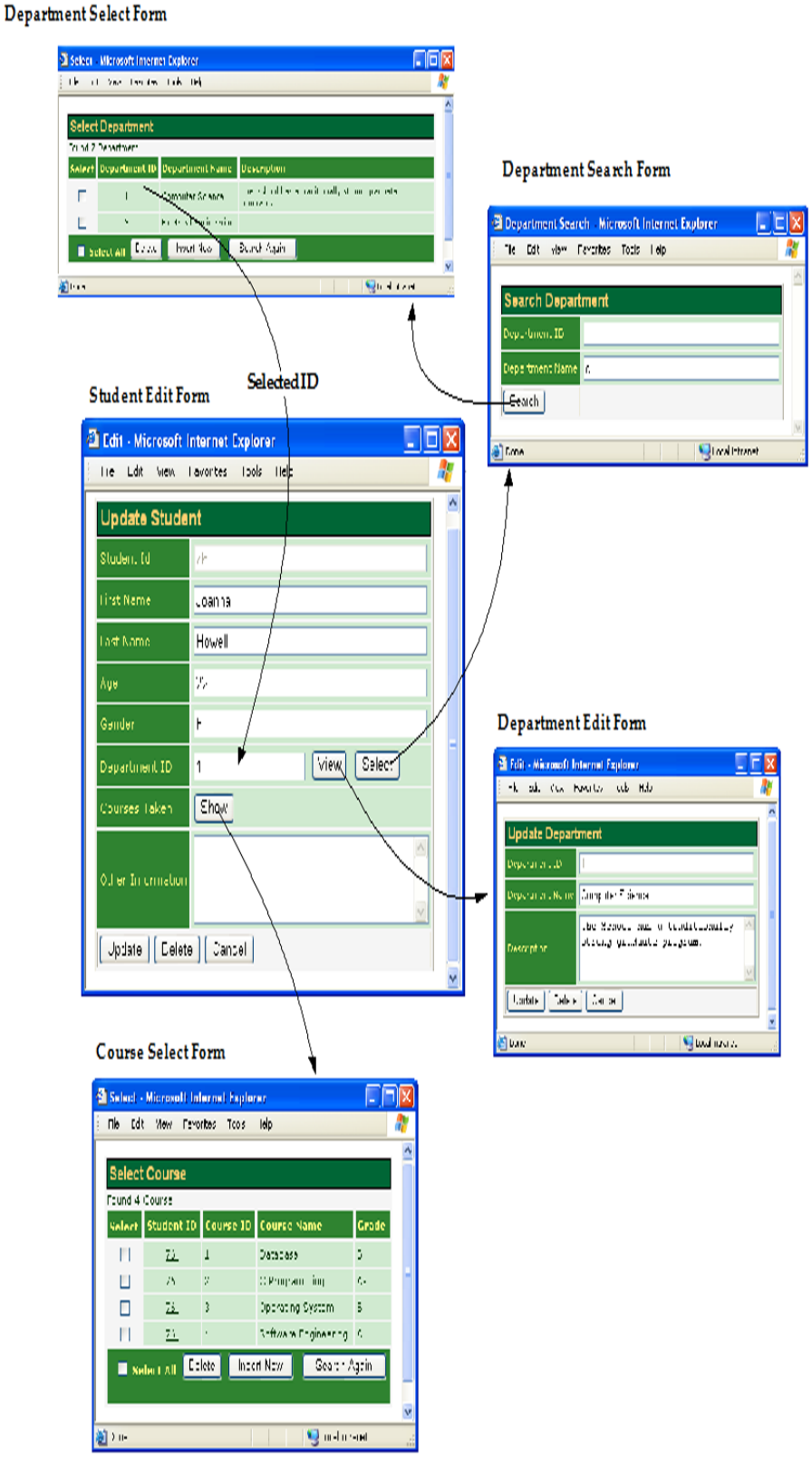


Figure 6: Student Edit Form

retrieved from the meta configuration file, table `geometry_columns`, and the map file. The default values of the parameters determined by the latter two are overwritten if they are customized in the meta configuration file.

A map-layer meta configuration file can define any parameters that are allowed for a map-layer configuration file. However, the default one generated by **map_mconfgen** contains for each map layer group the definitions only for the following parameters:

layer_selectable – A boolean option that determines whether the layer can be selected for spatial operations or not.

legend_label – The name of the layer that will be displayed on the map legend.

An example of a default map-layer meta configuration file is shown below:

```
$layer_groups = array(
   'grp_eo_py' => array(
      'layer_selectable' => true,
      'legend_label' => 'Eo Py',
),
   'grp_eosrc_' => array(
     'layer_selectable' => true,
     'legend_label' => 'Eosrc Pt',
),
...
```

The user may modify the values of these definitions and add others. For example, the meta configuration file can be customized as follows:

```
$layer_groups = array(
   'grp_eo_py' => array(
      'layer_selectable' => true,
      'legend_label' => 'EO Polygons',
      'search_script' => 'forms/eo/eo_py_eo_search.phtml',
      'select_script' => 'forms/eo/eo_py_eo_select.phtml',
      'edit_script' => 'forms/eo/eo_py_edit.phtml',
      'img_src' => 'images/eo_poly.png',
),
   'grp_eosrc_pt' => array(
      'layer_selectable' => true,
      'legend_label' => 'EOSRC Points',
      'search_script' => 'forms/eo/eosrc_pt_search.phtml',
      'select_script' => 'forms/eo/eosrc_pt_select.phtml',
      'edit_script' => 'forms/eo/eosrc_pt_edit.phtml',
      'img_src' => 'images/eosrc_pt.png',
),
...
```

The following customization was done:

1. `legend_label` was changed to the desired layer name for the map legend.
2. The names and locations of the search form, select form, and edit form are specified in `search_script`, `select_script`, and `edit_script`, respectively.
3. The source file name of the icon in the map legend was specified in `img_src`.

The order of the map-layer groups can also be changed to that desired in the map legend.

Then the map-layer configuration file shown previously can be generated by **map_confgen**.

WebGD Development History

The WebGD framework and WebGD-Gen were developed *incrementally* and *iteratively* during the last four years. We first implemented in 2000 an application that allowed point features to be inserted on a map by using ASP with ArcIMS and ArcSDE. In 2001, we re-implemented this application with ASP.NET as ASP.NET provides Web controls, which are better building blocks for Web pages. Based on this application, the first version of WebGD framework was created in 2002 in order to support multiple applications.(15)

In early 2003, we re-implemented an application called Motels Oregon with MapServer, PostGIS, and PostgreSQL(9) This version on Linux was more reliable and faster than the old one, as well as being built with free software. While implementing the next MapServer application, which was a germplasm resource management system (GEMGIS), we created the first version of WebGD framework for MapServer. This framework was then enhanced so that it can handle polygon features as well as point features.

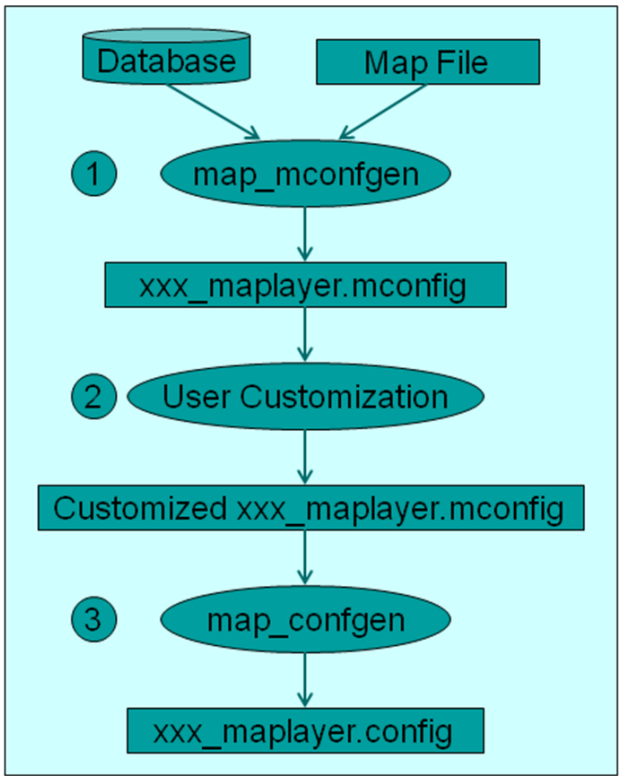


Figure 7: Process of Creating a Map-Layer Configuration File

The two major enhancements made to the WebGD framework in 2004 were *dynamic switching* of `spatial references` for different regions and *automatic generation* of Web forms that can be used to insert, query, and delete geographical features. Compared to an application that simply displays geographical features as points on a map, the current WebGD framework is roughly 20 times more complex in terms of the time we spent implementing the required features.

Conclusions and Future Work

We have developed the WebGD framework and the WebGD-Gen application generator for rapid development of Web-based GIS/database applications.

1. Geographical features, such as habitats of plants and animals, road-work sites, and waterlines, can be inserted, queried, and deleted with the map interface and Web forms displayed on a standard Web browser.
2. An application can be created without any programming. The map interface and Web scripts for data access can be automatically generated from configuration files, and those configuration files can be generated from the database schema and the GIS metadata, i.e. information stored in the table `geometry_columns`. Automatic generation of a Web-based GIS application not only reduces the development cost significantly, but it also facilitate incremental and iterative development of the application.
3. Dynamic switching of spatial references allows an application to cover different regions with different map files, projections, map legends, and quick-view lists. This is an important feature needed for an application that covers the entire USA or the world.
4. We created the WebGD framework by using only free open-source software. The software tools we use, such as the University of Minnesota MapServer, PostgreSQL DBMS, PostGIS, Apache, and PHP are all available for free. The GIS data used, such as those from USGS, TIGER/LINE, and Digital Chart of the World (DCW), are also in the public domain. Therefore, the framework is available for anyone for free use.
5. The cost of running our applications is extremely low. We could put copies of such large databases as Biotics, SSURGO2 soil data, and a part of National Germplasm Resource Information System on a $800PC.

Automatic code generation of a WebGD application will save a great deal of effort in the development of a spatial decision-support system. Although some manual customization is required, the time needed for customization can be lowered to weeks or months compared to the years required to build a *spatial decision-support system* from scratch.

The WebGD framework and WebGD-Gen are currently available upon request. In order to release them to the public as free and open-source software, we are looking for collaborators. We are also reimplementing the map interface by using OpenLayers

to support smooth panning.

Bibliography

[1] P. H. Dana Map Projection Overview http://www.colorado.edu/geography/gcraft/notes/mapproj/mapproj_f.html

[2] DM Solutions Group Inc. PHP MapScript http://www.maptools.org

[3] R. Kingston (1998) Web-based GIS for public participation decision making. In *Procs* of NCGIA PPGIS Meeting, Santa Barbara, California. Retrieved Map 2003 from http://www.ncgia.ucsb.edu/varenius/ppgis/papers/kingston/kingston.html

[4] Eum D. and Minoura T. (June 2003) Web-based database application generater. *IEICE Transactions on Information and Systems, Vol. E86-D, No. 6.*

[5] Fogelsong, C. (2002) Biotics 4.0 data model version 1.0. Retrieved January 5, 2004, from http://whiteoak.natureserve.org/hdms/HDMS-DataModel.shtml

[6] McKenna, Jeff MapServer PHP/MapScript Class Reference - Versions 3.6, 4.0 & 4.2 DM Solutions Group Inc.

[7] NatureServe (February 2002) Element Occurrence Data Standard. Retrieved January 4, 2004, from http://whiteoak.natureserve.org/eodraft/all.pdf

[8] NatureServe (December 2003) Biotics 4.0 Getting Started Guide. Retrieved January 5, 2004, from http://whiteoak.natureserve.org/hdms/biotics-learn-more.shtml (now obsolete).

[9] Sano J., Wanalertlak N., Maki A., and Minoura T. (July 2003) Benefits of web-based GIS/database applications. In *Prosc. of 2nd Annual Public Participation GIS Conference* Portland, Oregon.

[10] USDA-ARS Western Regional Plant Introduction Station, USDA - Agricultural Research Service, Pullman, Washington http://www.ars-grin.gov/ars/PacWest/Pullman/

[11] Ramsey, Paul PostGIS Manual Refractions Research Inc.

[12] University of Minnesota (2003) MapServer http://mapserver.gis.umn.edu

[13] Sharma, A. (December 2003) Web-based analysis module for a germplasm collection. Master of Science report, School of Electrical Engineering and Computer Science, Oregon State University

[14] Wangmutitakul P, Li L, and Minoura T. (March 2003) User Participatory Web-Based GIS/Database Application. *Proc. of Geotec Event Conference*

[15] Wangmutitakul Paphun et al. (2004) Framework for Web-based GIS/database Applications *Journal of Object Technology 3*, 209-225

[16] Wuttiwat T., Minoura T. and Steiner J. (May 2003) Using Digital Orthographic Aerial Images as User Interfaces *Proc. of ASPRS Annual Conference*, Anchorage, Alaska

Toshimi Minoura, Nirut Chalainanont, Junya Sano
Oregon State University
http://engr.oregonstate.edu/~minoura/research/creeda/
minoura AT eecs.orst.edu

db4o2D - Object Database Extension for 2D Geospatial Types

Stefan Keller

db4o stands for "database for objects". It's a native object oriented database management system (OODBMS) written in Java and .NET and thus targeted towards these two platforms. The software was first released 2001 by db4objects, Inc., and since then it got a major market share among the so called second generation object databases. It is available under two licenses ("dual licensing model"), an open source licence of type GPL for personal and non-commercial use as well as a commercial license.

In this contribution we first introduce db4o. In the chapter *OODBMS and db4o* we discuss the advantages, limitations and differences from a conceptual and a programmer's perspective. Then we report

about a student thesis project called db4o2D. Finally there is a conclusion with an outlook of the project db4o2D.

First steps in db4o

Let's start with some source code in order to give an idea how easy it is to persist application objects. Our first example demonstrates the four well known steps from database technology: Create, Read, Update and Delete (CRUD). The example is about Person objects which consist of last name, first name and year-of-birth:

```
// 1: Initialize an object container
ObjectContainer db= null;
try {
    // 2: Open db4o database (embedded mode)
    db= Db4o.openFile("addressbook.yap");

    // 3: Create and store some 'Kellers'
    db.set(new Person("Brian", "Keller", 1960));
    db.set(new Person("Clara", "Keller"));
    db.set(new Person("Test"));

    db.commit();

    // 4: Find and read all Kellers
    // with Query by Example
    ObjectSet result= db.get( \
        new Person(null, "Keller", 0));
    while (result.hasNext())
        System.out.println( \
            (Person) result.next());

    // 5: Update Clara's age
    result= db.get(new Person("Clara", "Keller"));
    Person found= (Person) result.next();
    found.setYearOfBirth(1970);
    db.set(found);

    // 6: Delete Test data
    result= db.get(new Person("Test"));
    while (result.hasNext())
        db.delete(result.next());

    // 7: Commit all
    db.commit();
} finally {
    if (db != null)
        db.close();
}
```

Fig. 1. Code fragment showing CRUD operations on person objects stored in an address book.

In this example we go though the following steps which show some CRUD operations:

1. The instruction following this comment initializes a container where db4o manages the set of objects to be persisted in a transactional way.
2. The second instruction opens a db4o database file. We propose to use the file extension .yap but this is not normalized. There is an official explanation which says that this is the abbreviation of "yet another protocol". Unofficially it's referring to Yap, a tiny island of Micronesia. As indicated we choose to use db4o in embedded mode. There exists also a client/server mode.
3. With the `set()` method of an `ObjectContainer` three newly created Person objects are stored. With `commit()` all created, changed or deleted objects are forced to be synchronized with the database file.
4. In this step we fetch all objects from the database which are equal to the last name "Keller" and iterate over the result. In this example "Query by Example" is used from the three query languages available from db4o.
5. When creating "Clara Keller" in step 3 we omitted the year-of-birth in the Person's constructor (since it's polite not to reveal the age of a woman beforehand). Now we insist in setting this value, so we have to look for objects like "Clara Keller" and update the first one the query gives back. We use again the set() method assuming that there exists a `setYearOfBirth()` method in the Person class definition.
6. For demonstration purposes we previously inserted also some test data which will be deleted in this code section. This time all objects retrieved by the query will be cleaned up with the `delete()` method of the `ObjectContainer`.
7. Finally we conclude with the `commit()` method and close the database.

The only definitions which are missing in this code fragment are the import statements as well as the Person class consisting of three constructors, the `setYearOfBirth()` method and optionally an overridden `toString()` method in order to print out Person objects nicely.

That's all to demonstrate the simplicity of a db4o enabled application. Next we explain some use cases of OODBMS, some additional features of db4o before an obvious extension is presented which will add geometry types to db4o.

OODBMS and db4o

Use Cases and Features

When there is only one application running on top of a database at a time and if it's a mobile application like in location based systems (LBS) or embedded software it's perhaps worthwhile to evaluate one of the so called second generation OODBMS. Because of the easy management of object relationships OODBMS are intrinsically well suited when complex object models, flat object structures or tall object trees are involved – which actually is often the case in Geographic Information Systems (GIS) and LBS.

These are some technical features of db4o:

- Embedded mode and client/server mode.
- No runtime server administration. Database properties are controlled out of the host application.
- Small space requirements of the program library on disk and in memory at runtime.
- Ease of use: db4o is using reflection application programming interfaces (API) from Java and .NET. So there are no extra annotations, no pre- or post-processing (byte code engineering), no sub classing nor interface implementation.
- Methods to control lazy loading of nested object relationships (depth).
- Replication tools as add-on.

Like one would expect from a database, db4o implements ACID (atomicity, consistency, isolation and durability) properties which guarantee reliable transactions: A transaction starts when opening or querying the database and ends with `commit()` and `rollback()` methods. Three approaches for queries are implemented: Query by Example, Simple Object Database Access (SODA Criteria) queries and "Native Queries". The first one was shown above. SODA queries are much like those used in object-relational (O/R) mapping frameworks. Some of the theory behind Native Queries stems from a project from Microsoft (LINQ 2007).

Advantages

Speaking of O/R mapping frameworks we have to compare OODBMS with this technique too. db4o is very fast according to benchmarks (Polepos 2007). A conceptual argument is that there is no object / relational impedance mismatch: No data types mapping and wrapping (unless wanted), no creation of separate relational schemas, no SQL dialects and no plain SQL query strings.

So it's important to mention that db4o offers decent support of agile software development techniques and refactoring: This is because of queries are written in the host language (Java, .NET) and thus are type safe. There are also nice schema evolution features and no SQL. Software engineers have an easier life than before because they reside in the object world as opposite to database professionals' world.

Limitations

db4o is probably not well-suited for being used in large data warehouses and in data mining. It's typically not recommended when several application are accessing the database with many views. One can see from the different query languages that there is no single standard and mature query language available compared to the pre-dominant SQL from the relational paradigm. Constraints like referential integrity are not (yet) part of any language except inside a Native Query which basically implements a call back function. Finally, the current lack of standardization was recognized. There are activities from the Object Management Group (OMG) in order to work towards a new release of the ODMG standard version 4.

Project db4o2D

PlaceLab (Intel 2006) — an Intel project — delivered mobile and desktop applications about a wireless LAN (WiFi) positioning. The database behind this software is a relational open source embedded database. A crucial component there is the management of access points together with their positions. This serves us here as a showcase of an object database which replaces the existing relational database and stores geometries as first class objects.

In a thesis project (db4o2D 2007) there was decided to use db4o and to adapt the broadly used Java Topology Suite library (JTS 2007). JTS follows the "Simple Features" standard (OGC 2006) which defines four 2D (2.5D) geometry attribute types: Point, LineString and Polygon as well as stable operations on it.

So in the following code fragment (c.f. Fig. 2) it is shown how wireless access points are created (2), stored (3) and reread (4). A point contains a coordinate value pair and includes a default coordinate reference system and measurement units.

```
// 1: Open db4o database (embedded mode)
db= Db4o.openFile("poidb.yap");
```

```
// 2: Prepare two JTS points
GeometryFactory factory= new GeometryFactory();
double latitude1= 47.225571, longitude1= 8.822271;
Point pt1= factory.createPoint(new Coordinate(
        longitude1, latitude1));
double latitude2= 47.225582, longitude2= 8.822282;
Point pt2= factory.createPoint(new Coordinate(
        longitude2, latitude2));

// 3: Create and store the access points
db.set(new AccessPoint( \
    1001, "802.11g", "wep", 7, pt1));
db.set(new AccessPoint( \
    1002, "802.11b", "open", 11, pt2));
db.commit();

// 4: Iterate over all access points
ObjectSet result= db.get(new AccessPoint());
while (result.hasNext()) {
    System.out.println( \
    (AccessPoint) result.next());
}
```

Fig. 2. Code fragment with wireless access points which contain point geometry.

Conclusion

The db4o2D project will become an add-on to db4o and is still ongoing. Although that the existing database access layer already was well separated it becomes obvious how smaller, simple and better to maintain the code becomes by using a pure OODBMS.

Future work includes a geospatial index and stress tests for large databases as well as for multi-threading client/server mode. Complex SODA queries is another open issue. It's noteworthy to state that this is no obstacle so far because Native Queries can be used in the meantime.

In order to disseminate object database technologies a non-profit group ODBMS.ORG (2007) was created. db4o is one of the leading object database projects and it tries to solve well known problems in embedded software, software engineering, LBS and GIS. Given that these problems have been around for quite some time one could say that second generation OODBMS like this are something like going back to the future.

References

db4o (2007): Homepage of db4o and db4objects, Inc.: www.db4o.com (last visited October 22nd, 2007).

db4o2D (2007): db4o2D project space, http://developer.db4o.com (last visited October 22nd, 2007).

Intel (2006): The PlaceLab Project, PlaceLab — A privacy observant location system, www.placelab.org (last visited October 22nd, 2007).

JavaWPS (2007): A positioning system in Java based on Wireless Local Area Network signals, www.gis.hsr.ch/wiki/JavaWPS (last visited October 22nd, 2007).

JTS (2007): Java Topology Suite, an API of 2D spatial predicates and functions, www.vividsolutions.com/jts/JTSHome.htm (last visited October 22nd, 2007).

LINQ (2007): The LINQ project, http://msdn.microsoft.com/netframework/future/linq (last visited October 22nd, 2007).

ODBMS.ORG (2007): A vendor-independent resource portal on object database technology, www.odbms.org, (last visited October 22nd, 2007). OGC (2006): OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture, www.opengeospatial.org/standards/sfa (last visited October 22nd, 2007)

PolePosition (2007): An open source database benchmark, www.polepos.org (last visited October 22nd, 2007).

Stefan Keller
Institute for Software and GISpunkt
University of Applied Sciences Rapperswil (UAS HSR)
CH-8640 Rapperswil, Switzerland
www.ifs.hsr.ch
Stefan Keller is professor for information systems teaching database technologies and programming.
stefan.keller AT hsr.ch

Google Summer of Code for Geoinformatics

Jan Ježek

Abstract

The GeoTools Referencing module has been becoming one of the most powerful tools focused on coordinate reference system transformations in the Java GIS world in recent years. The Referencing module in conjunction with the Coverage module presents a really strong tool for raster operations like reprojecting and transforming.

The aim of this paper is to describe new functionality that has been developed by author during the Google Summer of Code 2006 and 2007 projects. The usability of these new features will be discussed also with relation to the specific needs of reference coordinate systems that are used in the Czech Republic. Google Summer of Code itself will be also mentioned.

New functionality that was and still is being developed is focused on transformation methods based on interpolation. These procedures are usually applied in cases when transformation between coordinate systems is not some kind of exact mathematical relationship (such as cartographic projection or affine transformation for example).

This topic is closely related to rectification of old maps as well as the transformation of coordinate reference systems for those datums, that have been derived before GPS techniques started to rule and so their transformation into the global systems like WGS 84 is problematic and not as accurate as needed.

Google Summer of Code for Geoinformatics

Google Summer of Code (GSoC) is well known event that brings together students that are interested in open source software with core developers of projects from all branches. Huge projects like KDE, Ubuntu Linux, Apache Software Foundation, etc. participate in this program every year. The aim of GSoC is to get interested students involved in these project. Google plays the sponsor roll (among others) in the whole program and offer stipends for students that successfully participate. For detailed information about GSoC see (4).

The summer of 2007 saw the third volume of this event. The open source geospatial community started to participate in GSoC in 2006 when Refractions Research took the roll of mentoring organization. The projects that had been worked on were focused on the GeoTools library and the uDig desktop GIS. For more information about GSoC 2006 mentored by Refractions Research see (5).

OSGeo joined the program in the summer of 2007 and helped to get sponsorship for 12 students that contributed to a wide range of FOSS4G projects (GRASS, GDAL, GeoTools, Geoserver and uDig). For detailed information see (6).

Additional functions for coordinate system transformations in GeoTools and uDig

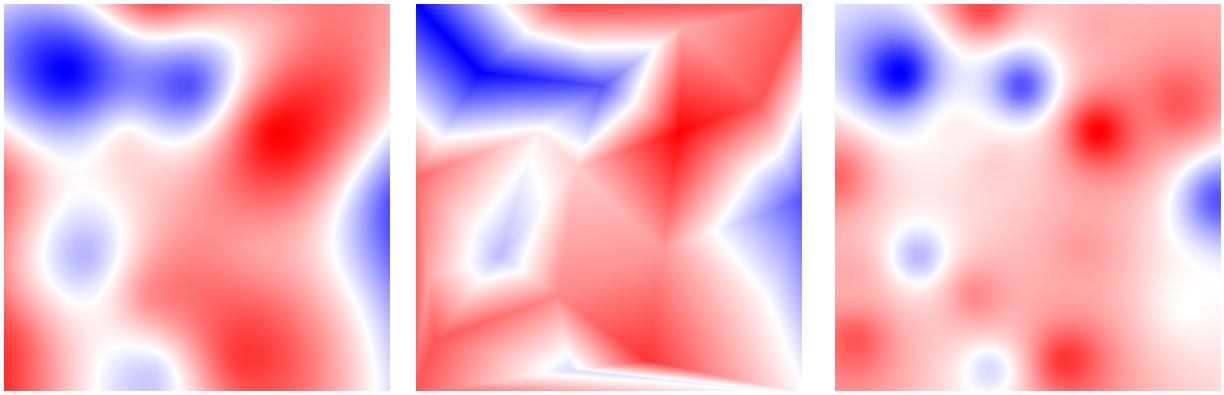
This part describes the work that has been done by the author during Summer of Code 2006 and 2007. The project was mentored by Jesse Eicher and Martin Desruisseaux.

The GeoTools Referencing package presents one of the most powerful tool for re-projecting and transforming in the Java GIS world. The package follows the OGC implementation specification (3). The package also offers a plugin mechanism, that lets users connect to persistent storage of datums and projection parameters such as an EPSG database (2).

The project that I've been working on during 2006 and 2007 has been focused on new coordinate system transformation algorithms. The aim was to implement tools that helps to solve the opposite transformation task — the task when we know some coordinates in source and target coordinate reference systems (mapped coordinates) and we are searching for the definition of transformation.

Linear transformations

There are a couple of transformation methods that can be unequally defined from known coordinates in source and target coordinate reference systems (CRS). This methods can be divided into two main



(a) Inverse distance weighted interpolation.

(b) Thin plate spline.

(c) Rubber sheeting.

Figure 1: Output of three interpolation methods

branches — linear transformation and no residual methods.

New implemented tools can be used to calculate transformation parameters for these linear methods:

- Similar Transformation
- Linear Transformation
- Affine Transformation
- Projective Transformation
- Position vectors (Bursa Wolf) transformation (3D similar 7 parameters)

These methods are unequally defined by different number of mapped position (Similar transformation needs 2, for example). When there are more coordinates than needed the least square method is used to minimize the square of distance between target and transformed source point. Currently there is just Cartesian distance taken into account.

No Residual algorithms

Another set of algorithms that has been implemented is focused on possibilities to calculate transformations that will exactly fit the source positions to target positions no matter how many mapped positions are defined (this is also called warp transformation). After studying the possibilities to define such methods through EPSG database conventions we choose the following approach.

One of the most general transformation that is defined in the EPSG database is the method based on a regular grid of coordinate offsets. Within this grid simple bi-linear interpolation is used so once you know the grid values you can apply the transformation quite fast. The family of grid-based methods includes:

- NADCON (EPSG dataset coordinate operation method code 9613) which is used by the US National Geodetic Survey for transformation between US systems
- NTv2 (EPSG dataset coordinate operation method code 9615) which originated in the national mapping agency of Canada and was subsequently adopted in Australia and New Zealand
- OSTN (EPSG dataset coordinate operation method code 9633) used in Great Britain

For more information see [(2)].

There have been 3 algorithms implemented that enable users to calculate the grid. These methods are:

Inverse distance weighted Interpolation: The offset values are calculated according to the distance from the known mapped positions

Thin plate spline interpolation: The name thin plate spline refers to a physical analogy involving the bending of a thin sheet of metal. In the physical setting, the deflection is in the z direction, orthogonal to the plane (1). The offsets in both direction (x and y or easting and northing) are calculated in this manner.

Rubber Sheeting method: The surface is divided into particular triangles by applying Delaunay's algorithm on the field mapped positions. Then the affine transformation on each triangle is applied. This method should be applied also as a piece-wise but calculation of the grid makes it more general, reusable an faster.

Implementation details

Described functions have been designed to become a part of the GeoTools referencing module.

The algorithms let you generate grid files that can be reused also in other software — in all that support EPSG methods explained above.The grid based transformation is designed to be performed using Java Advance Imaging (JAI) warp transform. This greatly helps improve performance especially when transforming raster datasets (JAI is using native library).

Other interesting results are when we try to visualise the calculated grids by converting calculated values to images. In this manner we can nicely see the distribution of the offsets and also the differences between applying particular method. You can see interpolated grids by all three methods using the same set of mapped positions in Figure 1.

All described functionality are currently located online.[5]

Migration into GeoTools 2.5 will take place during upcoming months. Other documentation and code examples can be found on the GeoTools website.[6]

uDig plugin

Finally, the simple uDig plugin that lets users calculate and apply described methods has been made. The plugin presents a GUI for accessing the described functions. First draft of this plugin is available from the community update site (see (7)) and lets you transform vector data using only a few methods (this was developed during Summer of Code 2006). A plugin that includes all described features has been developed only for trunk version of uDig, that is currently changing a lot so the stable version of plugin will be done after first milestone release of uDig 1.2.

Bibliography

[1] Wikipedia, the free encyclopedia, Thin plate spline, available at http://en.wikipedia.org/wiki/Thin_plate_spline , [29.10.2007].

[2] EPSG, Coordinate Conversions and Transformation including Formulas, available at http://www.epsg.org, [29.10.2007].

[3] OGC, Coordinate Transformation Service Implementation Specification, available at http://www.opengeospatial.org/standards/ct, [29.10.2007].

[4] Google, Google Summer of Code, available at http://code.google.com/soc/, [29.10.2007].

[5] Refractions Research, Refractions Research project ideas, available at http://code.google.com/soc/2006/refract/about.html, [29.10.2007].

[6] OSGeo, OSGeo project ideas, available at http://code.google.com/soc/2007/osgeo/about.html, [29.10.2007].

[7] Refractions Research, uDig update site, available at http://udig.refractions.net/confluence/display/UDIG/uDig+Update+Site, [29.10.2007].

Jan Ježek
h.jezek AT centrum.cz

[5]GSOC transformations page: http://svn.geotools.org/geotools/trunk/spike/jan/gsoc-transformations/
[6]GeoTools website: http://geotools.codehaus.org/New+Transformation+Algorithms+for+GeoTools+and+uDig

Topical Interest

A Generic Approach to Manage Metadata Standards

Julien Barde, Duane Edgington and Jean-Christophe Desconnets

Introduction

Informational Resources[7] (IR) management is a crucial part of environmental resources management. Indeed, the improvement of data processing and decision-making is strongly related to the ability to locate the relevant IR.

However, the exhaustive locating of relevant IR is a challenge for users as they face the following constraints:

- IR are *heterogeneous* (*language*, *semantic*, *syntax/-formats*, *metadata*, *access constraints* because of their *rarity* and *cost*...),
- IR are *distributed* into *heterogeneous Information Systems (IS)* whose interoperability first involves syntactic and semantic/spatial matching issues (answers to a natural language query often require its translation into as many queries as different kinds of IS).

Thus, the key issue to improve data retrieval is a better management of the *{metadata element, value}* pairs, which constitute any metadata sheet. Once aware of existing IS, the priority to locate the relevant IR is the management of the matching between the *heterogeneous metadata elements (syntactic)*, as well as between their *heterogeneous values (semantic)*. The global scale of environmental domains and the multidisciplinary context of related studies strongly increase these constraints and the need of *semantic and spatial referentials* management.

Metadata management

Metadata management is thus a priority before considering any data processing. Nevertheless, metadata management still faces the lack of referentials to homogenize: (i) the terminology of *metadata elements*, (ii) their *relationships* as well as (iii) their *values*. As a consequence, the quality of metadata management tools leans as much on the compliance with the reference standards specifications (syntactic: structure of metadata elements) as on the ability to use values from semantic referentials to edit instances.

The *heterogeneity of standardized metadata elements*

[7] An *informational resource* (IR) is the whole of data, information, knowledge produced, needed or treated by users (regardless of their format: hardcopy or digital...). According to the users, this term covers a report, a map, a picture, a video, a dataset, a data series, a database, a model....

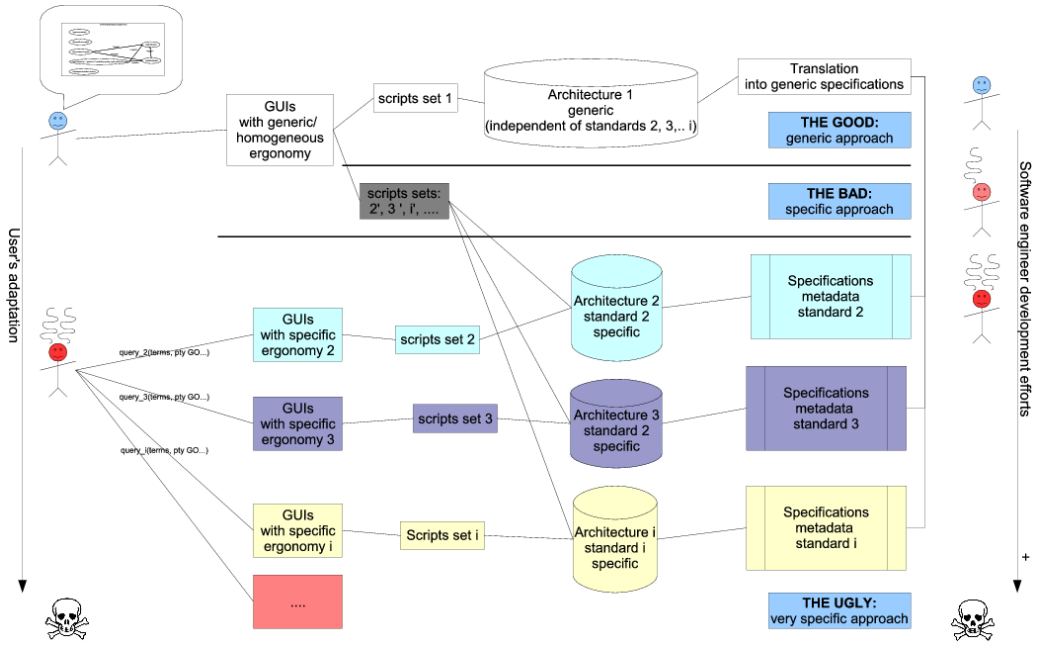


Figure 1: Benefits of a generic approach for multi-standards management

sets interferes with *IS syntactic interoperability*:

- standards often use similar *core metadata elements*, designating the same *concepts* by using different *terms*. These elements answer the following questions: *Where*? *What*? *When*? *Who*?... and are essential to retrieve IR,
- moreover, standards with *redundant scopes* generate wider *matching issues* since the same kind of IR could be described with different standards (like *FGDC* and *ISO 19115*),
- setting up new *international metadata standards* makes the previous national/local standards obsolete and brings *archiving issues* (potentially related to the previous *matching issues*),
- the recent use of *XML Schemas* to standardize their implementations decreases the redundancy of standards scopes and prevent wrong interpretations of their implementations (2). Standards can be used as referential types libraries (e.g. the case of *OGC* standards like the update of *ISO 19115* with *ISO 19139*, *SensorML* ...).

The *heterogeneity of metadata element values* interferes with *IS semantic interoperability* if values are not controlled (regardless of the chosen standard): the use of additional referentials is a key issue to improve IR descriptions and their retrieval by managing the *core metadata element values* (which are used in priority in most of the queries). Among them, the management of the following descriptions are crucial as they are the most complicated and ambiguous:

- *terminologic description* management with common (multilingual) controlled vocabularies/semantic referentials to valuate *"keyword" like metadata elements*. Moreover, these referentials help to set up *shared vocabularies* in *pluridisciplinary* contexts,
- *spatial description* management with shared geographic / spatial referentials facilitated by friendly GUIs improves the complex use of geographic information (*GI*) (in particular the use of formats like *GML*, *WKT*...).

The need of multi-standards metadata management tools is increasing. Indeed, even a single institute or project often has to manage more than one kind of IR.

Moreover, by considering users' and software engineers' tasks and needs to manage metadata, it appears that most of them are similar regardless of the standard implemented.

User's needs and tasks

According to their roles (administrator...), regardless of tools, *users tasks* to manage a metadata standard usually consists of:

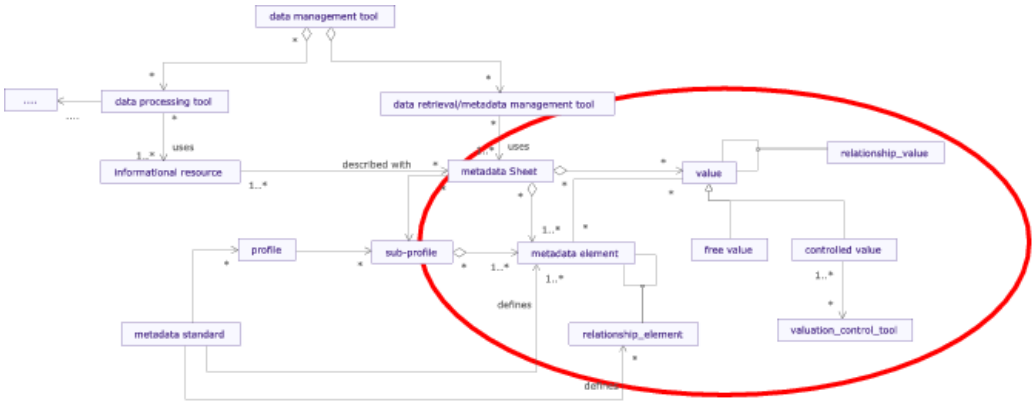


Figure 2: Generic expression of any metadata standard specifications

- *profiling* the metadata standards for their specific uses,
- *editing* standards instances to describe their IR (by relating values to metadata elements of the chosen profile),
- *locating* (and eventually *acquiring* according to access rights) the relevant IR for a given work by using a single *multi-criteria search engine* which allows sophisticated *spatial queries*,
- *import/export* metadata standards instances (usually XML),

Users need assistance to perform these tasks easily with friendly tools which are currently lacking. Most of the time, users express the need for single centralized access and tools with GUIs whose ergonomy is friendly and homogeneous from one standard to another (since tasks are similar). Indeed, heterogeneous software/IS implementing different or similar standards strongly increase user's accommodation efforts (as users have first to become familiar with each software to perform these tasks and then consider they are wasting their time). Finally, users need complementary components for any complex valuation process (*Web mapping tool*, *controlled vocabularies*, *calendar*...) with complicated format (like *GML*...). These use cases are illustrated with a UML diagram in the related slideshow (here).[8]

Software engineer's needs and tasks

In the same way, software engineer's main tasks and needs remain similar from one standard to another.

Software (engineers) tasks consist of:

- *satisfying user needs* by *complying with standards*,
- managing the *matching between core metadata elements* of different standards to answer basic queries efficiently,
- integrating and managing existing *semantic and spatial referentials* to warranty the quality of IR descriptions and to manage query expansion process. Indeed, an efficient data retrieval involves the management of as many queries as existing IS. Answers to these queries are all more difficult since terms and geographic objects used are heterogeneous,
- providing a rich *spatial data infrastructure* to manage and eventually process related IR thereafter.

Software engineers need to *minimize their development efforts* to implement metadata standards (7). They want to do so by answering similar user needs in the same (automated) way: by reusing a single script set and the same components (WMS...). This requires a generic approach (regardless of implemented standards) (5).

Generic approach vs. specific approach

Traditional specific implementations lead to heterogeneous data storage systems by translating directly specifications into physical heterogeneous data models and thus require specific scripts sets to process them (see illustration in figure 1).
For example, by using (manually or automatically) generated SQL from UML or XSD standards specifications, the resulting physical data models are going to be highly heterogeneous (as table and column names will match the metadata element names). Scripts to process their contents have therefore to be

[8]See UML diagram in slideshow: http://www.foss4g2007.org/presentations/viewattachment.php?attachment_id=46

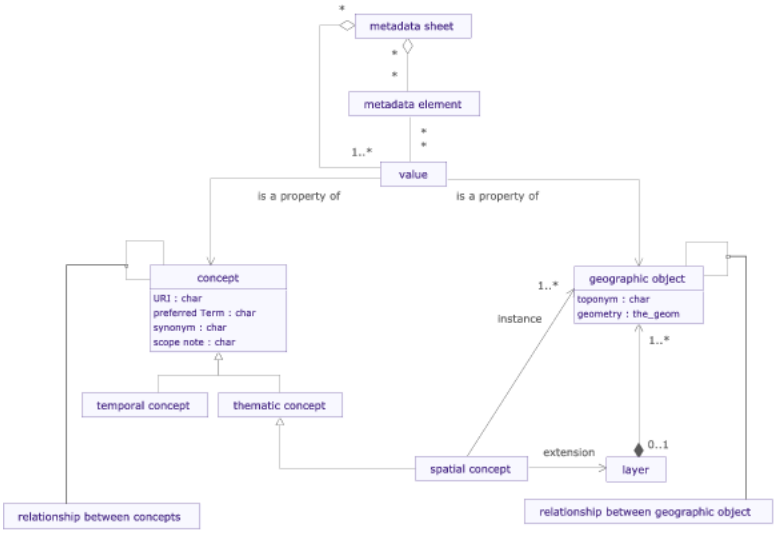


Figure 3: Properties and relationships of spatial and thematic concepts

adapted to these specific terms set to answer similar tasks. Script sets are thus heterogeneous from one standard to another.

According to the previous lists of tasks and needs, the figure 1 illustrates the benefit of a generic approach for both users and software engineers.

So far, existing tools don't cover these different needs as they mainly lean on specific approaches.

Generic models to manage efficiently {metadata element, value} pairs

We present in this section the ongoing generic models we are currently implementing to set up both a multi-standard metadata management tool and additional components which control the values of metadata elements by assisting the users at the same time. In particular these models will focus on the most crucial core metadata element values which are related to thematic and spatial descriptions.

A generic model to manage heterogeneous metadata standards

The goal is to design a generic pattern (or conceptual model as shown in figure 2) to describe any metadata standard and then set up a generic metadata management system which allows the control of essential values. We suggest expressing a standard as an *inventory of structured metadata elements* with potential additional tools to fill their *content* with controlled *values* (according to standards specifications and/or software engineer's will).
This approach is close to DOM's[9] goal which involves similar concepts to manage *nodes* and their *relationships* as well as their *content* in any kind of document. However, we only focus on the specific case of metadata standards.

Nevertheless, a standard rarely aims to control the potential values of the core metadata elements, and even more rarely relationships between values (in particular terms and geographic objects, date/period…). The control of such values is ruled by other specific standards. It is thus the role of the software engineer to integrate these standards by setting up complementary tools to manage these specific values.

We will thereafter focus on the specific case of the management of spatial and thematic values. We suggest a new model to manage their relationships.

A generic model to manage heterogeneous (thematic and spatial) values

The different kinds of {Metadata element, value} pairs are more or less crucial for data retrieval. In particular, certain values are especially difficult to control. Among them, *thematic* and *spatial descriptions* are both crucial as they are related to core metadata elements, involved in most of the users queries (re-

[9] Document Object Model

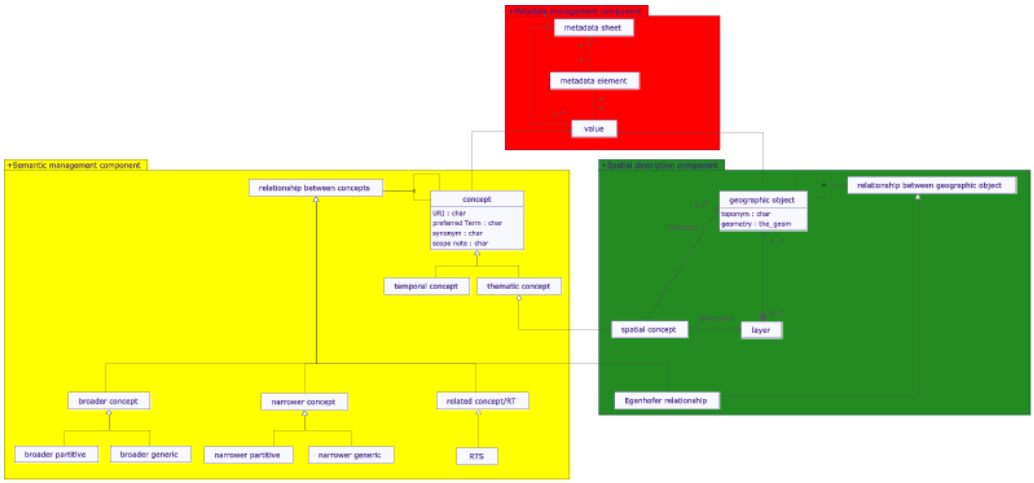


Figure 4: Summarization of the suggested generic approach

lated to *Where*? and *What*? criteria). We aim to manage them in a generic way by focusing on the user's intention: by focusing on the management of underlying *spatial* and *thematic concepts* (by using a formalization of their properties and relationships, see figure 3).

Indeed, the use of terms as values related to core metadata elements is often ambiguous:

- users often formalize their IR descriptions or queries by using such *terms*: *"swordfish, sea temperature, Madagascar, spring"*,
- semantic relationships management allows the system to relate different terms to expand these kinds of queries. For example by collecting other IR described with (*"Xiphias gladius..."*) which is a *synonym* of *"swordfish..."*) as it designates the same *concept* (in the same way as a picture and an image),
- the case of a *toponym* brings a new problematic as this kind of term could be both considered as a keyword or a geographical description. In fact, the geographic object related to the term/-toponym *"Madagascar"* could as well be designated graphically in a Web Mapping tool....

As illustrated in the figure 3, we suggest managing both *semantic and spatial relationships* between *thematic and spatial concepts* as well as *geographic objects* in the following way: "*a spatial concept as a kind of thematic concept whose instances are geographic objects*" (6). However it is important to consider that a *geographic object* is not necessarily related to a *term* or *toponym*.

The figure 4 summarizes the content of the previous generic models and give additional details to improve the management of both metadata elements and their values.

This model has been set up to be compliant with current reference standard implementations, for metadata, semantic and spatial information: standardized implementations of metadata standards (such as *XML Schemas*, *DTD*), (Web) Semantic standards (*SKOS* - related to *ISO 2788* and *5964* standards- */RDF/OWL*) and main GI standard formats. This generic model allows one to set up in a single architecture a *physical link between metadata elements and ontologies* to control their values (including spatial descriptions) and expand the queries efficiently.

In the same way, it is possible to set up additional controls for other crucial values: in particular *temporal* and *contacts descriptions* which answer the questions *When* and *Who*? Such control tools are usually *calendar* or *contacts directory* components (they manage date/period and human resources descriptions related to the IR).

The management of these additional referentials could be done independently of the metadata standards implemented. However, we aim to calculate the values of heterogeneous core metadata elements of the different metadata standards implemented in such a tool by using the same inventories of objects (managed in these referentials) as a basis for any standard. The management of these referentials in the same architecture facilitates the process. Thereafter by keeping track of objects used to describe IR in a dedicated *generic common index table* which duplicates the main descriptions (*What*, *Where*, *Who*, *When*...), it will be possible to answer effi-

ciently most of the users' requests, independently of the metadata standards used, by querying its records using richer values than standardized metadata element values (concepts URI instead of *terms*, 2D/3D geographic objects instead of *bounding boxes*...).

Model implementation with open source software

We present in this last section an implementation based on open source software.

Underlying technical choices

MDWeb is an open source product which is itself based on other open source software and standards. It implements this kind of architecture to set up a generic metadata management system. MDWeb:

- is a multistandard and multilingual metadata cataloging tool implementing a generic approach (like *M3Cat*, *MetaCat*...),
- is using a *three-tier* (client-server) architecture with:
  1. friendly *GUIs* (in Web browsers) with additional components (pop-ups) to assist metadata editing and searching:
     - *the spatial description* with Web Mapping tools which can be used as well to display the related GI: *Mapserver / Mapbuilder*,
     - *the thematic description* with Controlled vocabularies management GUIs to set up and browse of the-saurus / ontology: home made component.
  2. *applications scripts* (PHP/Javascript/XML with Apache Http server),
  3. *data storage*: *RDBMS* to manage metadata standards & spatial IR & related metadata & controlled vocabularies: *Postgres* with *PostGIS* (WMS for remote GI...). Import of *SKOS* files into Postgres by using *JENA* Java API. *XML* repositories.

Additional details on the main characteristics of the suggested three-tier architecture for the physical data infrastructure can be found in the related presentation (here).[10]

Examples of a possible generic GUIs set

By using *MDWeb* as a basis to implement this approach, it is thus possible to meet users and software engineer's needs, in particular by having a single set of homogeneous GUIs, regardless of the implemented standard (10):

- *Import of any new metadata standard* by translating formal specifications into the PDM[11] (for now only XML Schemas specifications import is automated),
- Set up of *profiles* of imported standards,
- *Metadata sheet edition* with additional GUIs to assist (automation, control...) thematic and spatial descriptions of IR (as shown in figure 5),
- Generic/*multi-standard search engine*,
- *Import/export* of standardized (usually XML) metadata sheet.

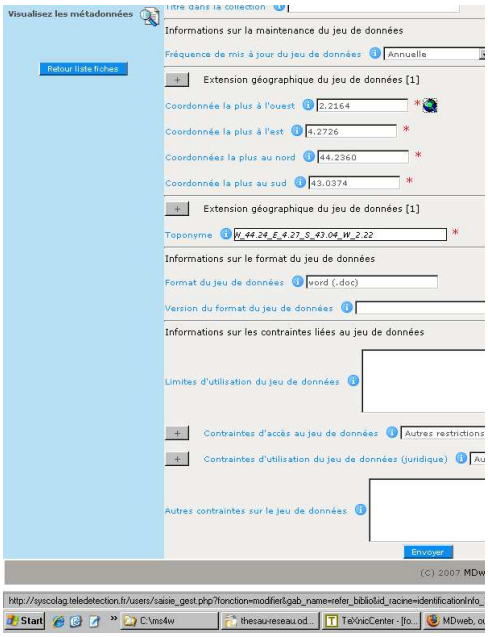


Figure 5: GUI to edit a metadata sheet

Conclusion and outlook

Data retrieval can be highly improved by managing metadata elements and their values in a better way. By implementing a generic approach (GUIs, scripts set, database) it is possible to manage into a single architecture :

- *heterogeneous metadata standards* (import, profiles, edition...),

[10] See presentation online at: http://www.foss4g2007.org/presentations/viewattachment.php?attachment_id=46
[11] Physical Data Model

- *heterogeneous values*: in particular *controlled terms* and *spatial descriptions* to describe *core metadata elements*,
- a *common index table* duplicating core metadata elements by using homogeneous values which can be used more efficiently by the *search engine* (no wrapper needed), especially to expand queries,
- *spatial IR described by metadata* can then be processed after being retrieved: either locally or remotely by using interoperable protocols or/and rich clients (*WMS*, *QGIS*, *uDig*...).

This kind of architecture is crucial to satisfy both *user's and software engineer's tasks and needs* by *minimizing adaptation and developments efforts* and by integrating the complementary tools to control crucial core metadata elements values.

Data retrieval is thus improved. In particular, by managing standardized semantic and spatial descriptions and their relationships in a common architecture, data retrieval can use *queries expansion* processes. It is thus possible to focus on specific use cases involving semantic and spatial relationships management like "find all the IR less than one mile of this geographic object (platform, sensor...) measuring the following physical parameter (temperature...)" by leaning on rich concepts, 2D or 3D geographic objects.... Moreover by using standardized semantic or spatial relationships (*W3C*, *OGC*...) the different kinds of queries can be exported and used in any kind of similar tool.

Generally, this implementation with an extensive use of *OGC standards and open source software* increases its ability to interoperate with external IS.

Bibliography

[1] L.M. Herborg, M.G. Bentley, A.S. Clare, S.P. Rushton (2003) The spread of the Chinese mitten crab (Eriocheir sinensis) in Europe; the predictive value of an historical data set. *Hydrobiologia* 503: 21-28.

[2] World Wide Web Consortium (W3C) (2004) XML Schema Part 0: Primer Second Edition. http://www.w3.org/TR/xmlschema-0/.

[3] World Wide Web Consortium (W3C) (2004) XML Schema Part 1: Structures Second Edition. http://www.w3.org/TR/xmlschema-1/.

[4] World Wide Web Consortium (W3C) (2004) XML Schema Part 2: Datatypes Second Edition http://www.w3.org/TR/xmlschema-2/.

[5] J. Barde (2005) Mutualisation de données et de connaissances pour la Gestion Intégrée des Zones Côtières. Application au projet SYSCOLAG. *Université Montpellier II* 285.

[6] J. Barde, J. C. Desconnets, T. Libourel, P. Maurel (2006) Generic conceptual models for data and knowledge sharing. Application to environmental domain. *Hydroscience and Engineering, ICHE 2006* 16: 407-420.

[7] Philip A. Bernstein and Laura M. Haas and Matthias Jarke and Erhard Rahm and Gio Wiederhold (2000) Panel: Is Generic Metadata Management Feasible? *The VLDB Journal* 660-662.

[8] Chad Berkley, Matthew Jones, Jivka Bojilova, Daniel Higgins (2001) Metacat: A Schema-Independent XML Database System. *SSDBM '01: Proceedings of the Thirteenth International Conference on Scientific and Statistical Database Management* 171. IEEE Computer Society

[9] Sergey Melnik, Erhard Rahm, Philip A. Bernstein (2003) Rondo: a programming platform for generic model management. *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data* 193-204. ACM Press

[10] V. Radha, S. Ramakrishna, N. Pradeep Kumar (2005) Generic XML Schema Definition (XSD) to GUI Translator. *Distributed Computing and Internet Technology* 3816:290-296. IEEE Computer Society

Barde Julien
Monterey Bay Aquarium Research Institute (MBARI)
http://www.mbari.org/staff/julien/
julien AT mbari.org

Towards Web Services Dedicated to Thematic Mapping

Abson Sae-Tang, Olivier Ertz

Introduction

Open standards favor interoperability of systems, and Open Geospatial Consortium (OGC) is the group that specifies the standards that allow geographic systems to interoperate. Among most known specifications, OGC defines the Web Map Server (WMS), the Web Feature Server (WFS), the Geographic Markup Language (GML), and the Styled Layer Descriptor (SLD) standards to solve the issue of spatial data sharing and interoperability. The project described in this paper puts the emphasis on the use of SLD to favor interoperability of geographic systems for thematic mapping.

What is SLD ?

SLD stands for Styled Layer Descriptor, it is an encoding that extends the Web Map Service specification to allow user-defined symbolization of feature data. It allows users (or systems) to determine which features or layers are rendered with which colors or symbols. SLD addresses the important need for users (and software) to be able to control the visual portrayal of the geospatial data. FOSS4G applications highly rely on and respect open standards, and SLD is implemented by Geoserver, Mapserver, deegree among many other software.

Next version of this standard is known as the Symbology Encoding Specification which is still in progress. The ability to define styling rules requires a styling language that the client and server can both understand. Symbology Encoding provides this language, while the SLD profile of WMS enables application of Symbology Encoding to WMS layers using extensions of WMS operations.

Is SLD ready for thematic mapping ?

Concretely, SLD is a useful and complete specification for styling your maps. For each layer you could say "color all my line features in blue", or "make all polygon borders black, and the insides pale yellow", or even "use little triangles for all my points". But you can also define even more complex styles. You can define the style rules based on attributes of the features in a layer. In a roads data set, you can style highways with a three-pixel red line, style four-lane roads in a two-pixel black line, and style two-lane roads in a one-pixel black line, thanks to an attribute that indicates the type of road.

But is it as good for thematic mapping? That is, for choropleth maps (Figure 1a), proportional symbols (Figure 1b), overlaid symbols (Figure 1c), juxtaposed symbols (Figure 1d), pie charts (Figure 1e), bar/histogram charts (Figure 1f), etc. These charts mainly make use of style rules based on feature attributes to transform statistical data to a graphical representation on the map. So, this project is first a study on how far we can go with SLD for thematic mapping.

With the current specification of SLD, it's more or less possible to describe thematic maps, but there are some drawbacks :

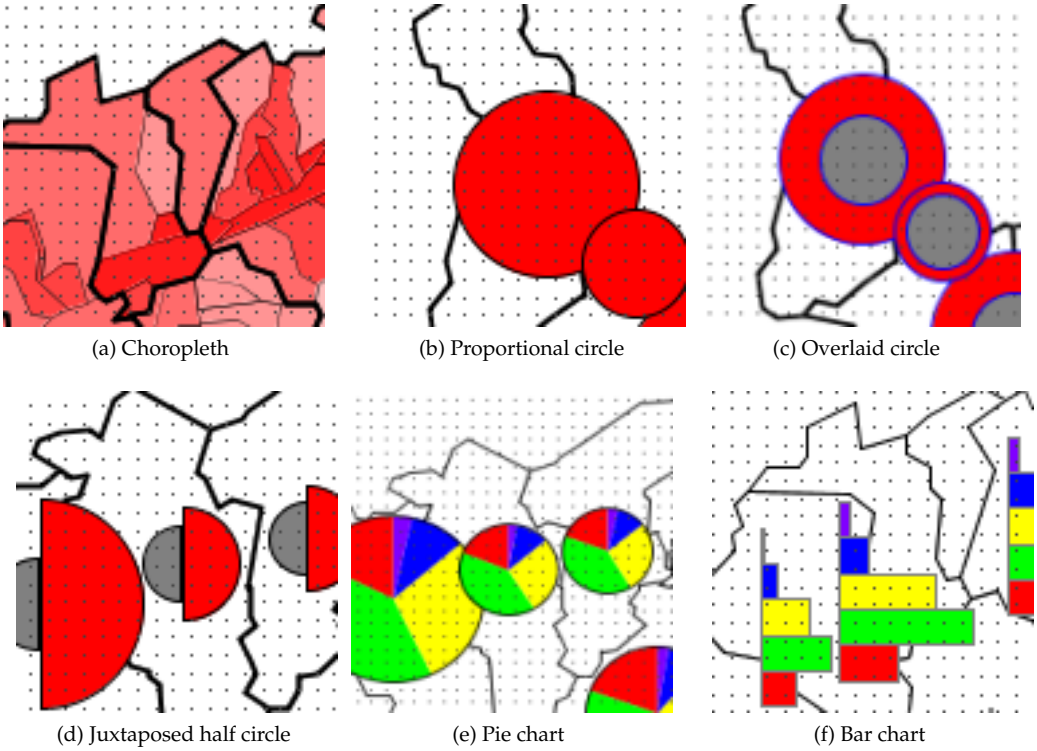
Choropleth map : one rule with a filter (class boundaries) per class, each rule having its polygon symbolizer with the fill color to apply.

Proportional symbols : a point symbolizer with a built-in graphic mark like circle, a fill color, and its size controlled by a data attribute.

Bivariate symbols : a mix of the filter and color of a choropleth map, and the point symbolizer of proportional symbols. Both size of the symbol and fill color are controlled by two data attributes.

Overlaid symbols : two rules with a filter for the rendering order (the fact that the smallest symbol has to be in front of the greatest). And two point symbolizers per rule, each with its size controlled by a data attribute.

For juxtaposed symbols, pie chart, and bar/histogram chart maps, it starts to be more complex and even unpleasant to describe them with SLD. Concretely, how do we hang two juxtaposed symbols on a unique point or centroid? Same for slices of a pie chart or bars of a histogram. One solution would be to use InlineFeature (from SLD 1.1) to draw the symbols, bars of the histogram for example. But unfortunately, InlineFeature uses GML to create temporary features, not graphics! So this is not accept-



(a) Choropleth (b) Proportional circle (c) Overlaid circle

(d) Juxtaposed half circle (e) Pie chart (f) Bar chart

Figure 1: Thematic maps

able. The second idea would be to use a kind of third party application for producing pictures of the symbols to represent points with external graphic links. Nonetheless if this is more acceptable, it is too verbose. Because one rule with a filter per symbol is required (given 50 points, SLD will need 50 rules).

Towards an extension dedicated to thematic mapping

This initial study on how far we can go with SLD lead us to the idea of extending the symbology encoding of SLD. We call it SLD-T (even if it has nothing to do with WFS-T). Such an extension wants to extend the grammar in order to introduce specific terminology the thematic mapper is used to, ease the description of thematic maps, and reduce verbosity and redundancy.

Basically the idea is to create an abstract ThematicSymbolizer (like a PointSymbolizer is) that will be include in the SLD schema by extending the existing Symbolizer element from a Rule, with specializations for each kind of thematic map (Figure 2).

CategoryThematicSymbolizer : for maps with classifications (i.e. choropleth and bivariate map). This symbolizer is built-on ThematicCategory elements to describe the classification type (by unique value, by range value, etc.).

SimpleThematicSymbolizer : for maps without classification (i.e. proportional symbols). It is a simple wrapper of "standard" symbolizers to let them inherit useful generic elements from the ThematicSymbolizer like symbol priority and placement (see MultiThematicSymbolizer).

MultiThematicSymbolizer : to depict several thematic symbols per feature (i.e. overlaid and juxtaposed symbols). A BaseSymbolizer is used to group common rendering element (like Stroke or WellKnownName) and avoid redundancy. For overlaid symbols, the rendering process order is managed by the Priority element which can be controlled by a feature attribute. For juxtaposed symbols, the PointPlacement (inspired from the TextSymbolizer) allow to define an AnchorPoint and a Displacement for each symbol.

ChartThematicSymbolizer : for chart symbols (i.e. pie and bar charts). A ThematicMark (following the idea of graphic Mark) is used to specify the chart type (pie or bar). And a ChartParts

element to describe the bars and slices composing the complex chart. As all ThematicSymbolizer can be rotated, one can also create an histogram.

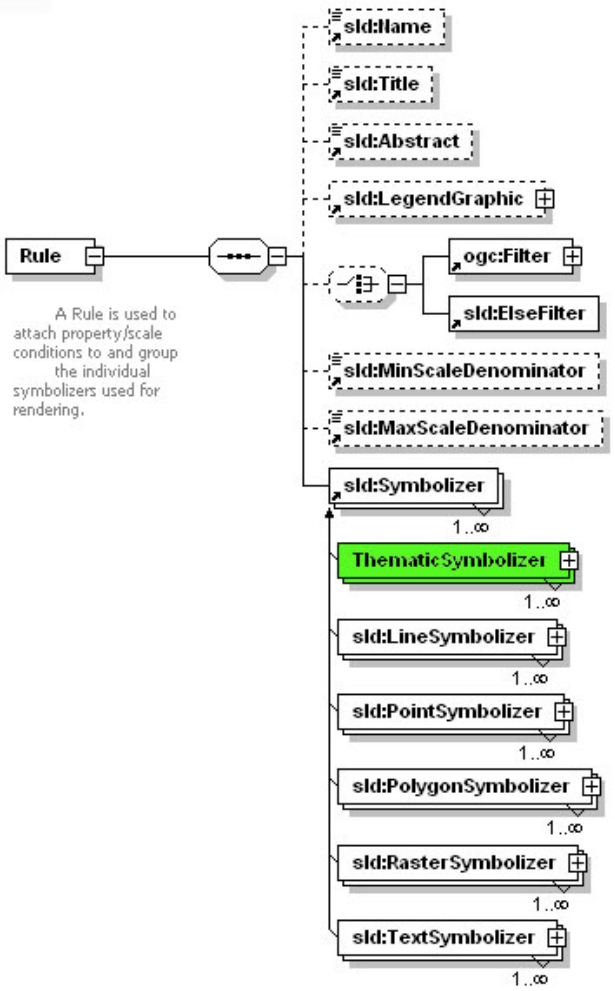


Figure 2: Rule with a new symbolizer

This is a brief summary about the extension. Complete XSD schema and document specification with examples are available online.[12]

SLD specification was originally meant for WMS. The user can define a SLD document and send it to a WMS server, and it returns the layer after applying the style you provided. But it's also useful as part of an OWS compliant desktop application. It could take a SLD file and apply it to a WFS response in GML that it receives. So, it makes sense to use such a styling specification server-side and client-side.

Consider this basic scenario: a cartographer or thematic mapper designing a nice and meaningful map on its SLD compliant desktop application, and as soon as the map is ready, he pushes it on a WMS server to share it. He will push the data and the styling. If interoperability is first about sharing data, it is finally about sharing maps with the appropriate styling for visualization, and the user wants also to share thematic maps.

Conclusion

The initial study of this project tries to depict how far and how to use the specification for thematic mapping further then just for basic styling. Moreover, to enable SLD for complex thematic mapping, the project describe a solution with an extension of the symbology encoding. The extension has the aim to introduce specific terminology thematic mapper is familiar with, ease the description of thematic maps, and reduce verbosity and redundancy the use of "standard" SLD could produce.

As a proof of concept, a first implementation has been started on GeoTools, only about juxtaposed symbols. It was quite easy, because of an initial strong support of SLD and rendering model. The advantage of using GeoTools is that the library is used server-side for Geoserver but also client-side with uDig. But there are many FOSS platforms on which a complete implementation could be done, and no decision has been made. Future plans include an implementation but also a real use case. Notice, to have a really complete implementation, the rendering of legend graphic is mandatory.

SLD is more then ever a key element for Spatial Data Infrastructure, and its acceptance is probably crossing a step as we see more and more projects building SLD renderer and especially editor to ease user's life to create SLD (GeoServer, Mapbender, OpenLayers, etc.). Finally, at the so called "web thematic mapping"' side, it is good to see MapServer now able to create complex thematic maps with pie and bar charts, and also client-side application like CarThema5 (based on gvSIG), JumpChart (based on JUMP), OrbisGIS (based on GeoTools and GDMS) are ready to go for thematic mapping. This is all good news going the right way.

Acknowledgements

The authors wish to thank their partners at Micro-GIS SA and Camptocamp SA for their collaboration

[12] XSD schema and document specification: http://geosysin.iict.ch

on parts of the research behind this paper.

Bibliography

[1] A. Sae-Tang, and O. Ertz, *Towards Web Services Dedicated to Thematic Mapping*, HEIG-VD, IICT/geo.SYSIN http://geosysin.iict.ch.

[2] OGC Styled Layer Descriptor Specification, http://www.opengeospatial.org/standards/sld.

[3] OGC Symbology Encoding Specification, http://www.opengeospatial.org/standards/symbol.

[4] M.D. Teixeira, R. de Melo Cuba, and G.M. Weiss, *Creating Thematic Maps with OGC Standards Through the Web*, CPqD Telecom & IT Solutions, http://www.gmldays.com/papers/Teixeira.html.

[5] M.A. Manso, A. Maldonado, R. Hernandez, D. Ballari, and J. Moya, *GEOSISMO : Visualization of Events and Seismologic Characteristics in the Internet*, Madrid Polytechnic University, http://redgeomatica.rediris.es/ICA_Madrid2005/papers/manso.pdf.

Abson Sae-Tang
HEIG-VD IICT/geo.SYSIN
IICT/geo.SYSIN, http://www.iict.ch
musy-abson.sae-tang AT heig-vd.ch

Olivier Ertz
HEIG-VD IICT/geo.SYSIN
IICT/geo.SYSIN, http://www.iict.ch
olivier.ertz AT heig-vd.ch

Interoperability for 3D Geodata

Experiences with CityGML and OGC Web Services

Hans Plum and Dr. Markus Lupp

Summary

Storage, processing and visualization of 3D geodata are an important subject in the GIS world even before the leading search engine introduced its globe viewer. Usage of standards of the Open Geospatial Consortium (OGC) open up new possibilities for combination and usage of 3D geodata. First practical experiences show promising results.

Introduction

Processing and visualization of 3D geodata became a common subject during the last years. Some indicators for this are the number of offered software solutions but also the amount of interest for the development of CityGML. CityGML is a GML-based exchange format for three dimensional digital city models, that is already implemented in a number of software products. With the definition of CityGML and application of OGC Web Services for access to and visualization of 3D geodata the areas of 3D geodata processing and Spatial Data Infrastructures (SDI) are converging.

This article is discussing solutions that were realized using technology from the deegree project. The mentioned projects are: “Storage and administration of 3D city models for the cities of Bonn, Berlin and Hamburg”, “Visualization of digital terrain models for the Federal Agency for Cartography and Geodesy of Germany”, “Realization of a transactional CityGML WFS for the Open Geospatial Consortium” are outlined.

OGC-Standards with relevance for 3D

A number of discussion papers and specifications of the OGC are of importance for 3D geodata handling. In particular these are CityGML as data model and exchange format, Web Feature Service and Web Coverage Service for data access and Web Terrain Service and Web 3D Service respectively for visualization purposes.

City Geography Markup Language (CityGML)

CityGML defines a semantic object model for 3D objects in urban areas. It is a GML application schema that is it model objects of an application domain using constructs of the Geography Markup Language. In this aspect CityGML is a semantic model as well as an exchange format.

CityGML is so far mainly developed by a working group of the SDI initiative of Northrhine-Westfalia, although members from all over Germany are part of this group. In version 0.3 CityGML was introduced into the OGC and published as a discussion paper (1). CityGML 1.0 will in short time become an official OGC Best Practice paper.

Web Feature Service

A Web Feature Service (WFS, (2) allows to query geodata modeled in GML. Filter Encoding (3), an SQL-like language encoded in XML is used to query a WFS. A WFS that allows not only to read, but also write access (create, update and delete) is called a transactional WFS (WFS-T).

WFS is an official OGC-standard in the current version 1.1.0. A WFS implementing this 1.1.0 specification has to support GML 3.1.1 – the same version that is the base for CityGML. It is therefore possible to use a WFS as a data access layer to CityGML.

Web Coverage Service

A Web Coverage Service (WCS, (4) allows to access all kinds of data that is modeled "field-based", e.g. Raster- or TIN-based. Examples of such data are those created by remote sensing or digital terrain models. In the context of 3D SDI a WCS can be used to access terrain models. WCS is an official OGC standard with the current version 1.1.0.

Web Terrain Service

A Web Terrain Service (WTS), still in OGC discussion paper status, generates Views of 3D scenes. In contrast to a WMS that creates 2D visualizations, an image depicting 3D data is generated.

Unfortunately, the development of the WTS specification advances rather slow. The current draft version bears the name "Web Perspective View Service" (WPVS) to express that the service is able to depict 3D objects besides "Terrain".

Figure 1 shows the result of a GetView-request. A digital terrain model is depicted that is textured with aerial photographs. On top of the terrain a number of buildings are displayed (one of them transparently).

WPVS creates presentations of 3D objects. The most important operation of this service is GetView which returns static pictures of 3D landscapes. The GetView operation can be seen as an extension to the GetMap operation of WMS. In comparison to GetMap, GetView defines additional parameters allowing to specify a 3D scene. Among these parameters are a rotation angle and the azimuth of the depicted scene. As the result of a GetView operation is a (static) image; it is not possible to navigate directly through the scene. A WPVS client is therefore in comparison with real 3D viewers not very interactive, but can be implemented as a web application using DHTML without the need for browser plugins. Another advantage is that such a simple and web-based 3D client can easily be integrated with other web-client software, like e.g. WMS-based portals.

The challenge when creating a WPVS client is to hide the complexity of a GetView request behind an easy to use graphical user interface, that allows navigation in 3D space.

Use cases

To support the projects mentioned in the introduction, the following use cases have to be supported.

Storage of digital city models

Digital city models are often created using CAD systems and stored in CAD file formats. This results in a number of disadvantages. It is not possible to easily select parts of the city model or to organize updates. Because of this reason, organizations who own such city models need homogeneous data that best is stored in a database.

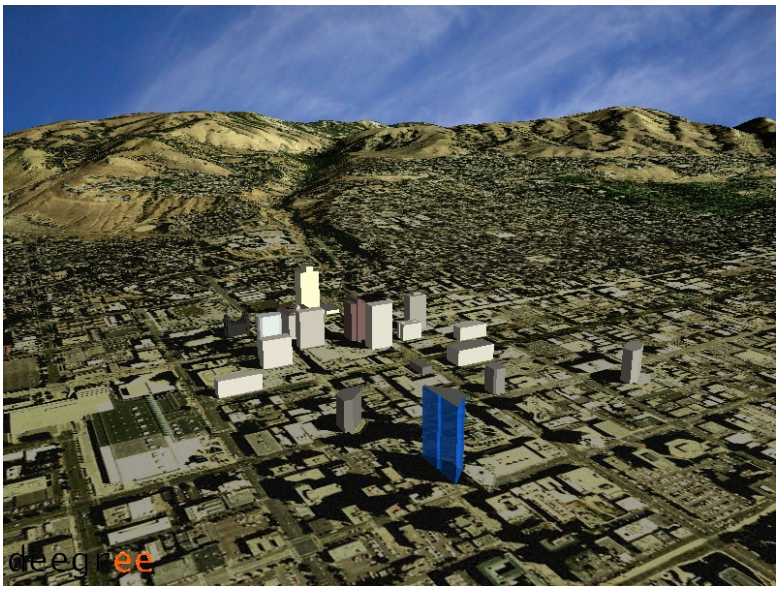


Figure 1: Visualization of terrain and buildings using deegree WTS/WPVS

To support this use case it is necessary to store CityGML in a – most likely relational – database. For access to this database a WFS is the obvious choice, CityGML can then directly be inserted and pulled out of the database.

To control the access to the WFS it is necessary to use an access control mechanism. In the mentioned projects components of deegree iGeoSecurity are used for this.

Web Visualization

The advantage of 3D geodata is mostly to be found in its possibilities on visualization. Application areas are support of urban planning processes and navigation. In the context of planning processes, 3D geovisualizations allow to display the consequences of planned projects before they are realized.

Tourist information systems can also benefit from 3D visualization. Recognition of landmarks or navigation can be enhanced. Great potential also lies in the coupling of classical 2D maps with 3D scene visualizations.

For marketing purposes of the data itself, terrain and city models are displayed on the Internet. The potential of the data is shown in this way.

Architecture of a 3D SDI

In the following, the architecture of 3D SDIs realized by deegree will be described (c.f. figure 2).

The building information models will be kept in a spatial enabled database, e.g. PostGIS or Oracle Spatial. A transactional WFS (WFS-T) supports the access for reading and writing of the city models. In order to control the information flow, especially transactions against the WFS, a owsProxy is used to protect the building information against unauthorized access. The editing component – mostly a CAD system – accesses the data via owsProxy and WFS.

Digital elevation data can also be saved in a geodatabase. Especially for TINs or points. Alternatively, raster data can be saved in the filesystem. In order to support a fast access method the mechanisms described above have to be used. Accessing digital terrain data in raster format are provided by a WCS. Getting the according terrain model while rendering a city model is easily done through the Web Coverage Service.

On the right hand side of figure 2 the visualization process is shown. deegree-WPVS accesses the data out of the geodatabase. Furthermore, it is possible to integrate external WFS- or WCS-services. Textures like ortho imagery or maps for navigation are also needed. Via a WMS these kind of data can be integrated. A web-based WPVS-Client provides a graphical user interface that can be used in web browsers.

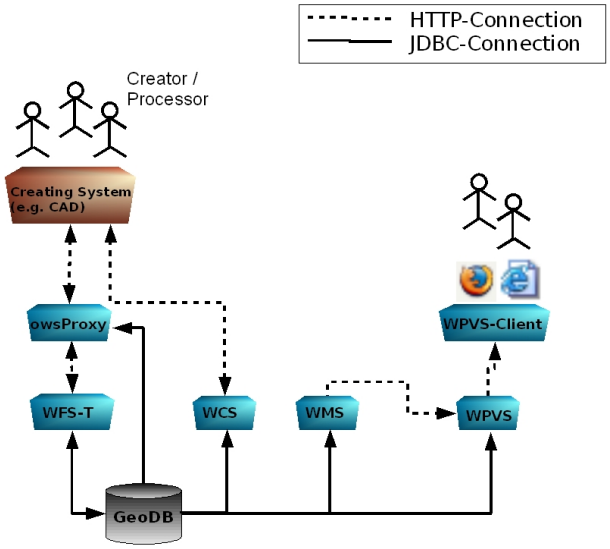


Figure 2: Architecture of a 3D SDI

Conclusion

The development of CityGML defines an important step towards 3D SDIs. The experiences using deegree components for development of such systems that were made in a number of projects are promising. They show that it is already possible to create 3D SDIs using Open Source software.

The mentioned components are available via www.deegree.org. At the time of writing the RC1 for WPVS (including a client), WFS and WCS are available as easily installable WAR archives.

Bibliography

[1] Gröger, G., T. Kolbe and A. Czerwinski (2006) City Geography Markup Language. OGC project document 06-057r1. https://portal.opengeospatial.org/files/?artifact_id=16675.

[2] Vretanos, P.A. (2004) OpenGIS Web Feature Service Implementation Specification Version 1.1.0. OpenGIS Project Document 04-094. http://www.opengis.org/specs/?page=specs.

[3] Vretanos, P.A. (2004) OpenGIS Filter Encoding Specification Version 1.1.0. OpenGIS Project Document 04-095. http://www.opengis.org/specs/?page=specs.

[4] Evans, J. (2003) OpenGIS Web Coverage Service Specification Version 1.0.0. OpenGIS Project Document 03-065r6. http://www.opengis.org/specs/?page=specs.

Hans Plum, Dr. Markus Lupp
lat/lon GmbH, Bonn, Germany
http://www.lat-lon.de
plum|lupp AT lat-lon.de

A Model-Driven Web Feature Service for Enhanced Semantic Interoperability

Peter Staub

Introduction

This article addresses current research issues in the field of interoperability of heterogeneous GIS. We focus on heterogeneity at the level of conceptual data models. The presented research project of a model-driven Web Feature Service aims at enhancing semantic interoperability. The approaches of data interoperability such as OGC web services (OWS) are combined with methods of model interoperability. The model-driven approach of conceptual data modelling allows for keeping data models independent from any particular system.

Interoperability is a crucial capability to deal with in the context of geospatial applications and information communities. The use of web services is well-established and useable in a standardised way due to the efforts of the OGC. However, OWS such as the Web Feature Service (WFS) (3) provide data interoperability, but no model interoperability. Conceptual model mappings are a precondition for semantic interoperability but are not supported by OWS.

Among European initiatives for geodata infrastructures – such as INSPIRE[13] – the need for interoperability not only on the data level, but also on the model level, grows. The research project described in this article was initiated in the context of a project in the Lake Constance region[14]. The mentioned project aims at creating a cross-border web-based GIS for applications.

In the presented research project, we introduce a **model-driven WFS** (**mdWFS**) which combines both the advantages of OWS for data interoperability and those of the model-driven approach for conceptual data modelling. Furthermore, formalism for establishing conceptual model mappings is developed and a prototype is implemented. Because of this combination, the mdWFS we introduce is an approach that provides enhanced semantic interoperability.

Fundamentals of Data Modelling and Semantic Interoperability

The Model-Driven Approach

The main idea of the model-driven approach is to describe (geo-)data models using a conceptual schema language (CSL). The use of a CSL for modelling allows for keeping data structures independent from any particular system or transfer format such as XML or GML. Virtually any transfer format can be derived from the conceptual schema (syn. model) automatically – given an adequate model compiler.

If you want to reach semantic interoperability, you will have to create conceptual model mappings. A conceptual model mapping is converted into mapping functions $\mathcal{F}_M$ from a source schema A to any target schema B:

$$A \xrightarrow{\mathcal{F}_M} B$$

The model-driven approach consists of four steps (see figure 1):

1. Specification of an *application domain* (i. e. "what we are talking about")
2. Specification of a *CSL* with a coherent UML metamodel
3. Description of the application domain with the chosen CSL → *conceptual schema*, platform independent model (PIM)
4. Derivation of any format schema (e. g. a GML Application Schema) → *logical and physical schema*, platform specific model (PSM)

As mentioned above, we assume that the generation of the logical schema is *automatically* carried out by a compiler and the encoding is done by an adequate encoding program.

In the presented research project, the (textual) CSL *Interlis* is applied for data modelling. Interlis is a Swiss standard (8) and is widely applied in cadastral and planning applications. Interlis is based on a UML 2 profile and a compiler[15] generates XML schemas (Interlis format) or GML Application Schemas from any given Interlis data model.

[13] INSPIRE project website: http://www.ec-gis.org/inspire/index.cfm
[14] Bodensee-Geodatenpool (Lake Constance geodata pool) project website: http://www.bodensee-geodatenpool.net
[15] Interlis compiler: see http://www.interlis.ch. The compiler is free and open source.

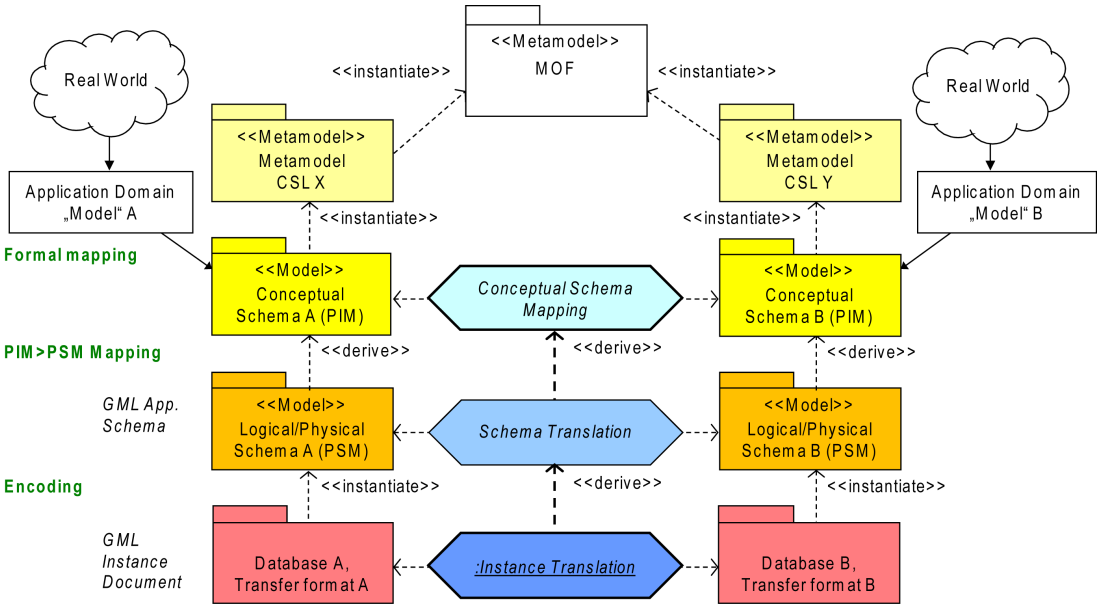


Figure 1: Model-driven approach and semantic interoperability

According to the Model-Driven Architecture (MDA) specified by the Object Management Group (OMG) (4), the generation of a format schema from a conceptual schema is referred to as a PIM–PSM mapping. In terms of mapping directions, the PIM–PSM is also called a "vertical" mapping, whereas model mappings for semantic interoperability are "horizontal" (i. e. PIM–PIM) mappings (see next section).

Semantic Interoperability

Technically, there are two main aspects characterising "interoperability": 1) *Data interoperability* = the ability of a system or process to exchange datasets using certain data transfer formats. 2) *Model interoperability* = the ability to establish conceptual model mappings in order to execute semantic data transformations.

To achieve semantic interoperability, different data models have to be mapped. A translator then interprets the mapping rules from the conceptual model mapping and executes the instance translation automatically.

Semantic transformation approaches can be classified as follows (1):

- *Level of abstraction:* Semantic transformation can be performed on different levels of abstraction (on the conceptual level, on the logical level and on the physical (i. e. transfer format) level). A semantic transformation on the conceptual level is platform independent, whereas approaches on the logical or physical level are platform specific.
- *Orientation:* Horizontal semantic transformation between different schemas on the same level of abstraction (PIM–PIM; PSM–PSM) vs. vertical semantic transformation between different levels of abstraction (PIM–PSM).
- *Level of automation:* Creating mapping rules by hand vs. automated schema matching which is only partially practicable.

Shortcomings of Existing Approaches

One possibility is to integrate all data into one central system. This is very costly and requires expert knowledge. In order to integrate the data into the central system, 1:1-format conversions have to be carried out. This is often lossy because a data format which is different from the original one is in general not able to express the entire semantics of the original data format. Besides that, the inevitable redundant data storage possibly causes outdated data.

Existing OWS such as the WFS have some shortcomings with regard to semantic interoperability: OWS allow for syntactic interoperability (i. e. data interoperability) but not for semantic interoperability (i. e. model interoperability). Conceptual models of source systems are hidden from target systems and semantic transformations are not supported. So, the

WFS lacks in the ability to handle conceptual model information aside from data information.

Concept of the Model-Driven WFS

Preconditions for a Web-Based Semantic Transformation

If we want to have a web service that allows for data interoperability *and* that is able to store and deliver conceptual schemas, a number of preconditions must be fulfilled. It must be assured that conceptual schemas are described (i. e. modelled) using a textual CSL with its graphic representation in UML 2 (and the respective exchange format XMI). Furthermore, a formal language is needed for expressing schema mapping rules on the conceptual level of abstraction. Finally, we use a standard WFS interface to provide satisfying data interoperability.

Web Service Requirements

Web-enabling semantic transformations means in our case actually designing a web service. This service has to comply with two main requirements:

1. Provide access to geospatial data based on the data's original conceptual schema (source model) and on any user-defined conceptual schema acting as the target model.
2. Interoperability with existing OWS.

The mdWFS Interface

We designed a service called "model-driven Web Feature Service" (mdWFS) taking these requirements into account. The mdWFS has the task to store and deliver conceptual schemas and to carry out semantic transformations (PIM–PIM mappings) by means of interpreting transformation models. After a semantic transformation, the mdWFS configures a standard WFS to provide a service for data interoperability. The standard WFS is configured according the target model but delivers transformed feature data from the source model.

WFS Protocol Extensions

In order to create a WFS that is able to store and deliver conceptual schemas, the OGC WFS specification needs to be extended. In the mdWFS specification, the extensions described below are applied (1):

- To provide a service protocol for the mdWFS, a new request parameter `SERVICE=mdWFS` is implemented.
- The `GetCapabilities` request is extended to provide a `SchemaList`. This list includes each conceptual schema that is available in the service.
- The `DescribeFeatureType` request is extended to provide the `XMI` format for transferring model information.
- Finally, a whole new request `DoTransform` is introduced. This request transfers the conceptual mapping schema to the mdWFS and invokes the semantic transformation.

UMLT, a Conceptual Schema Mapping Language

Concept of UMLT

We introduce a conceptual mapping language that can be used to create conceptual mapping schemas (syn. transformation schemas) for semantic transformations. This formal language must comply with several requirements in order to be useable. Transformation schemas must be comprehensible also for non-computer scientists. Therefore, a UML 2 metamodel as well as syntax for a human useable textual notation (HUTN) is developed. Transformation schemas are represented in visual form (UML activity diagrams), in textual form (derived from Interlis CSL) and XML (i. e. XMI), respectively. Common standards in the field of data modelling are taken into account[16]. We also apply an abstraction layer for (geo-)data types.

Two existing approaches from the OMG were examined. First, the Meta Object Facility Query/Views/Transformations formalism (MOF-QVT) (5): this formalism is designed for the transformation of metamodels, e. g. UML→Java. MOF-QVT models are hard to understand and their visual representation helps little. The standard is complex since it actually consists of three languages: Relations, Core and Operational. Furthermore, the MOF-QVT standard is predominantly applied for PIM–PSM implementation mappings.

Another approach that was examined is UML 2 Activities. UML 2 activity diagrams can be used to describe transformations in terms of activity sequences. A clear description of the semantics and of the transfer format (XMI 2.1) is provided in the Su-

[16] Such as standards from OMG, OGC and ISO

perstructure for UML models. UML 2 models are comprehensible and a number of implementations and open source APIs are available.

Because of the above considerations, our conceptual mapping language is based on an independent extension of the UML 2 metamodel. To specify the language elements, a UML 2 model is created and the textual notation of the language is defined by a set of EBNF grammar rules. At project stage, we call our conceptual mapping language "**UMLT**".

UMLT Language Elements

The language elements of UMLT are an inheritance of UML 2 Activities (7). We introduce the following language elements (see figure 2):

- `StructuredTransformation`
- `SelectionCriteria`: selection of input data through a logical expression.
- `VirtualAssociation`: manage input objects that are not actually associated with an association object. These input objects may have link attributes or foreign key attributes that are evaluated at runtime in order to get calculated relations[17]. During a semantic transformation, such objects can be associated *in a virtual way* if needed. The `VirtualAssociation` is introduced (in contrary to a common "derived association") to provide a means to explicitly specify the join property of the association with the `joinCriteria` expression.
- `TransformationAction`: inheritance from a UML `OpaqueAction` providing an activity element which cannot be structured any further. This is a transformation's elementary action.
- `AssignmentDefinitions`: address primitive types or expressions as value specification.
- `MappingRule`: the actual object mapping. Built as a composition of assignment definitions.
- `AssociationBinding`: selecting associated input objects, one may define how these associations are evaluated during input.
- `JoinType`: an enumeration type to specify the join type of the association binding.

Prototype Implementation

In the context of the presented research project, a proof-of-concept prototype is implemented. Besides the WFS protocol extension and the UMLT language specification, this prototype consists of a model parser, a mapping model editor and a prototype test bed. The model parser and the editor are developed in the Eclipse environment[18]. The model parser creates an XMI file from a UML/Interlis data model and also from a UMLT mapping model.

In the prototype test bed, we use an ORACLE Spatial database and a deegree WFS implementation[19] on which the mdWFS is built upon. Figure 3 shows the steps of a semantic transformation using mdWFS.

We primarily focused on the WFS extension and on the conceptual mapping language UMLT. We consider ORACLE Spatial as a very suitable RDBMS for our needs, providing powerful spatial features. Therefore, we use the RDBMS we already had at hand although it is not a FOSS solution. Principally, an mdWFS can be applied on any RDBMS with a spatial extension.

Before you can start working with mdWFS, you need to configure the database according to the source data model A. This can be done using an existing FOSS tool called "ili2ora"[20]. This tool allows to configure an ORACLE Spatial database according an UML/Interlis-data model and to import feature data into this database.

1. Client B sends a model-catalogue request to the mdWFS
2. The mdWFS provides a catalogue of available data models
3. Client B chooses a source data model (i. e. model A) and orders the model information
4. The mdWFS fetches model information A and sends the model (XMI) or a model reference to the client B
5. Client B creates the model mapping $M : A \xrightarrow{\mathcal{F}_M} B$ by specifying adequate UMLT mapping rules
6. The transformation model *and* the target model B are parsed and translated into XMI and sent to the mdWFS in a `DoTransform`-request
7. According to the target model B, the mdWFS configures an ORACLE Spatial database using ili2ora again

[17] A different example is a geometry/topology relation: a building *on* a parcel
[18] Eclipse: http://www.eclipse.org
[19] deegree project page: http://www.deegree.org
[20] Source: http://www.eisenhutinformatik.ch/interlis/ili2ora/

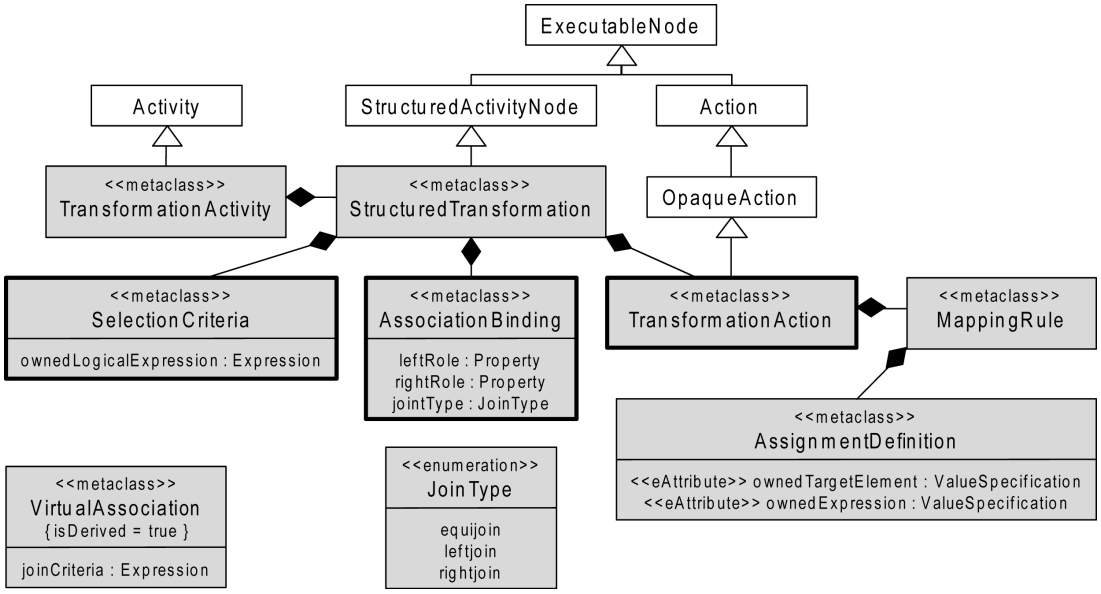


Figure 2: UMLT language elements

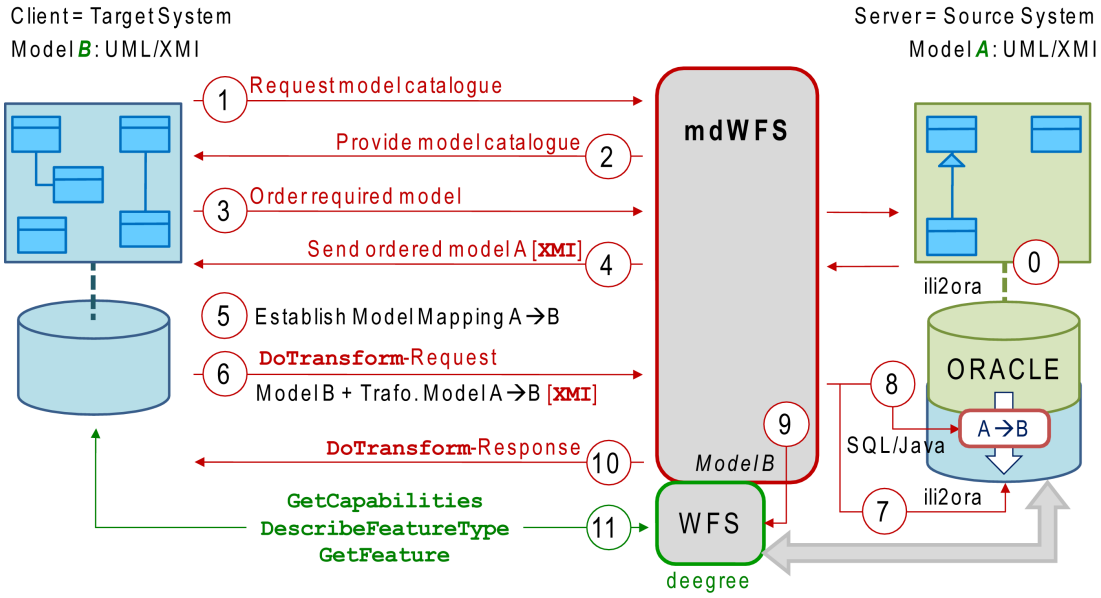


Figure 3: Prototype implementation test bed

8. The mapping rules from the transformation model are translated into SQL statements and Java instructions in order to actually transform feature data from source/server model *A* into target/client model *B*
9. The mdWFS configures a standard WFS (deegree) according to the target model *B*. This WFS is connected to the database containing the transformed feature data
10. After finishing the transformation, the mdWFS sends a `DoTransform`-response to the client *B*
11. Client *B* accesses the transformed feature datasets from model/database *A*, transformed into model structure *B*, via standard WFS requests.

Conclusions

The current evolution of GI systems shows that a conceptual schema language is usually applied for geodata modelling. This is a necessary precondition for semantic transformations on the conceptual level. Any given application domain can be characterised by different data structures. This leads to different data models. Therefore, conceptual model mappings must be established in order to achieve semantic interoperability.

The new *mdWFS* presented in this article implements the methodology of the semantic transformation at the conceptual level of abstraction what allows for a much enhanced semantic interoperability.

Potentially, the mdWFS can be integrated in other (OWS based) infrastructures due to the sound basis of GI standards that are applied.

Acknowledgements

I would like to thank my colleagues at the ETH Zurich and at the TU Munich for their company and good collaboration in this research project: Dr.-Ing. A. Donaubauer, H. R. Gnägi, A. Morf, F. Straub. I also appreciate the excellent project lead of Prof. Dr. A. Carosio, ETH Zurich and Prof. Dr.-Ing. M. Schilcher, TU Munich.

This project is co-funded by the German Federal Office for Cartography and Geodesy BKG and the Swiss Federal Office for Topography swisstopo.

Bibliography

[1] A. Donaubauer, F. Straub, M. Schilcher (2007) mdWFS: A Concept of Web-enabling Semantic Transformation. Proceedings of the 10th AGILE Conference on Geographic Information Science, 2007, Aalborg.

[2] H. R. Gnägi, A. Morf, P. Staub (2006) Semantic Interoperability through the Definition of Conceptual Model Transformations. Proceedings of the 9th AGILE Conference on Geographic Information Science, 2006, Visegrád.

[3] OGC Open Geospatial Consortium (2005) Web Feature Service Implementation Specification: 1.1.0. OpenGIS implementation specification OGC 04-094.

[4] OMG Object Management Group (2003) MDA Guide Version 1.0.1. OMG specification omg/2003-06-01.

[5] OMG Object Management Group (2005) MOF 2.0 Query/Views/Transformations Specification. OMG specification ptc/05-11-01.

[6] OMG Object Management Group (2005) MOF 2.0/XMI Mapping Specification, v2.1. OMG specification formal/05-09-01.

[7] OMG Object Management Group (2007) UML Unified Modeling Language: Superstructure, version 2.1.1. OMG specification formal/2007-02-05.

[8] SNV Swiss Association for Standardization (2006) INTERLIS Reference Manual, version 2.3. Swiss standard SN 612031.

Peter Staub
ETH Zurich, Institute of Geodesy and Photogrammetry, GIS Group
http://www.gis.ethz.ch
peterstaub AT ethz.ch

Spatial-Yap: A Spatio-Deductive Database System

David Vaz and Michel Ferreira

Introduction

The paradigm of Deductive Databases extends traditional databases with deduction abilities. Knowledge is not only represented extensionally[21] but also through intensional logic rules. A typical approach in building such deductive database systems is to couple a logic programming system with a relational database system.

The original query language, Datalog (1), restricted attribute data to ground atomic values, such as numbers and strings, which were the typical data stored in databases. However, current databases store much more structured data, such as the geometric attributes of spatial relations.

In this paper we describe the extension of the Yap Prolog (2) compiler, a free, open-source logic programming system, in order to handle spatial data, providing a state-of-the-art solution for its modeling, querying and mining. A proposal of extending Datalog to Spatial Datalog has been described in the literature, in the framework of Constraint Databases (3). The approach followed in Spatial-Yap is different and closer to the spatial databases community, as it is based on spatial terms, rather than on polynomial inequalities. Spatial-Yap can build spatial logic terms from vectorial data in spatial relations, and provides a highly declarative programming environment for its handling, supported, for instance, by the natural specification of recursion, inherent to topological relationships, and by a powerful ADT, as is the logic term. Although the current focus in Spatial-Yap is much more on declarativity than on efficiency, the system is able to explore advanced features of Yap, such as a Prolog to SQL translator and a tabling engine based on tries, to improve performance.

Logic Programming and Inductive Logic Programming

Logic programming (LP), of which Prolog is the canonical language, is an attempt to implement Colmerauer and Kowalski's idea that computation is controlled inference (4). The motivation for the LP paradigm is to separate the specification of *what* the program should do from *how* it should be done. This was summarized by Kowalski's motto:

$$algorithm = logic + control.$$

Prolog programs use the logic to express the problem and rely on the Prolog system to execute this specification. Prolog implements a subset of first order logic known as Horn clause logic. A Prolog program is a set of relational rules of the form:

`A :- B1, B2, ..., Bn.`

meaning: `A` is `True` if `B1` is `True` and `B2` is `True` ... and `Bn` is `True`. These rules are given a procedural interpretation which reads as:

to solve(execute) `A` solve(execute) `B1` and solve(execute) `B2` ... and solve(execute) `Bn`.

The precise procedural interpretation used in the execution of Prolog programs is a restricted form of SLD-resolution (5).

Deductive database systems are database management systems which are also designed around a logic model of data and whose query language is a set-oriented version of Prolog, known as Datalog. Database relations are naturally thought of as the *value* of a logical predicate and the high expressive power of logic expressions is used to query such relations. The deductive part of such systems comes from the fact that the logic programming engines take *intentions* (or comprehensions), which express properties, and are able to materialize these intentions in *extensional* knowledge (relational tuples or facts).

This process of deduction, which goes from intentions to extensions, is computationally much simpler than the reverse process of going from extensions to intentions. However, being able to derive an intentional representation from extensional data, inferring a general rule from examples, is also crucial. This is the goal of a logic programming paradigm known as Inductive Logic Programming (ILP) (6).

ILP is a research area formed at the intersection of Machine Learning and LP. ILP systems develop predicate descriptions from examples and background knowledge, thus deriving an hypothesized logic program which entails all the positive and

[21] Definition of terms: http://en.wikipedia.org/wiki/Extensional_definition

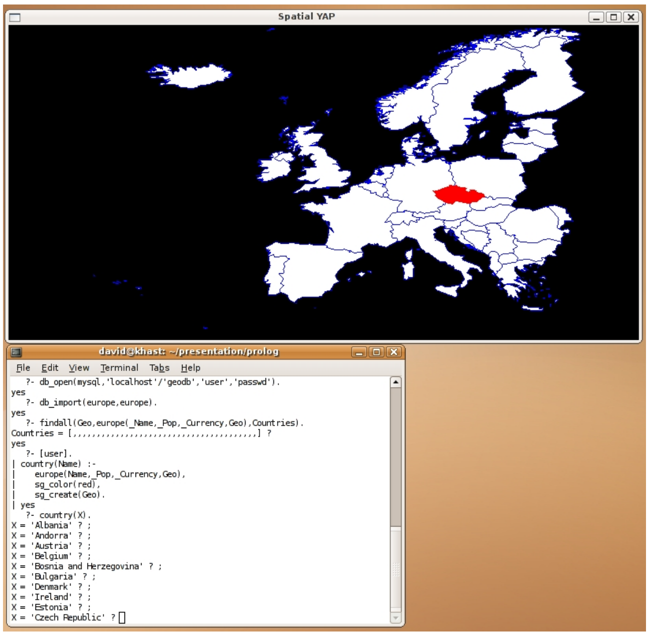


Figure 1: Spatial-Yap example.

none of the negative examples. To derive a theory with the desire properties, many ILP systems follow some kind of *generate-and-test* approach to traverse the *hypotheses space*. An important characteristic of the ILP approach to data mining is that it is multi-relational, being able to formulate theories which involve data in several relations, while many important data mining techniques are only able to look for patterns in a single relation. This is particularly useful for spatial data mining, which is inherently multi-relational (or multi-layered).

A Spatio-Deductive Database System

Spatial Yap results from a sophisticated interface between several components. The two main components are the Yap Prolog system and MySQL RDBMS, which are coupled through the MYDDAS interface (7) (Mysql/Yap Deductive DAtabase System). This interface is responsible for coupling these two systems, as illustrated in Figure 2. MYDDAS transparently translates logic queries into SQL statements, implements the conversion into Yap terms of MySQL attributes and explores the YapTab tabling engine for solving recursive queries involving database goals. The level of sophistication of this interface is very high, with the fetching of relational tuples being implemented directly in WAM choice-points, supporting pruning operators (8).

To build the spatial deductive database system we extended the MYDDAS interface to support MySQL Geometry Types. Two more components are fundamental to build Spatial Yap: a spatial operators library based on the well known GEOS library and a visualization component.

When dealing with spatial data it is essential to be able to graphically represent such data. Representing a map as a set of Prolog terms is visually unacceptable, from the point of view of a GIS user. Even more important is the representation of a spatial operation, such as the intersection of two polygons, in a graphical way. User-driven spatial analysis and the representation of spatial queries result sets require a visualization component to be added to any spatial database system. Here we could have used one of the existing FOSS, such as MapServer, but we rather needed something simpler that worked as a graphical spatial top-level, tightly coupled to the text top-level of our Prolog system. Interaction between these top-levels was our main goal, rather than sophisticated graphical display. Figure 1 show a screenshot of the interaction between the two top-levels of Spatial-Yap.

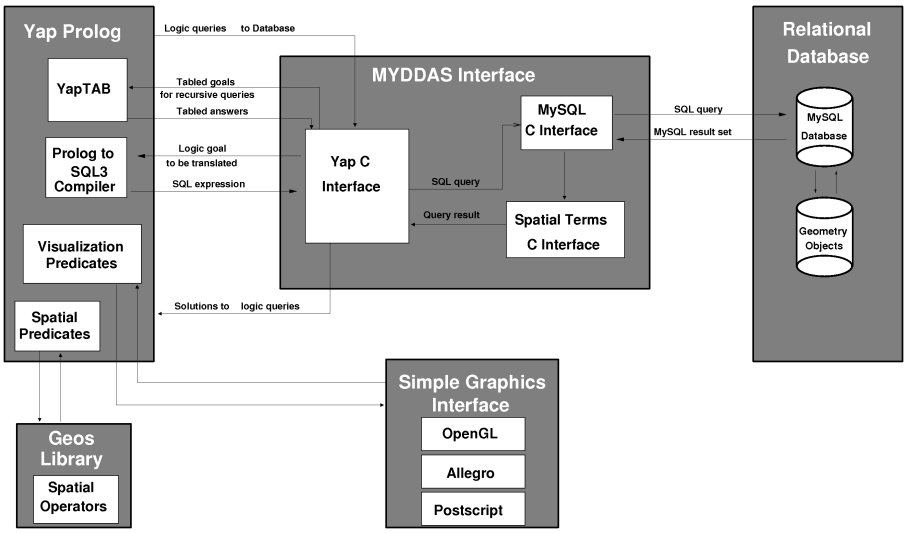


Figure 2: Spatial Yap Blueprint.

Current Applications

In this section we describe two undergoing projects where we are applying Spatial-Yap.

Study of Traffic Behaviour in the City of Porto

The aim of this project is to study traffic behavior in the city of Porto, second largest in Portugal, with a road network totaling 965 kms, shown in Fig. 3. We are interested in understanding factors affecting traffic, not only time and day related, but also including intrinsic geographic entities, such as the presence of a school in a street segment and its influence in traffic congestion time-slots. More ambitious goals include the automatic derivation of a road signalization layer, including traffic lights and stop signs locations, based on mobility patterns, or the inference of likely destinations of drivers that can automatically activate navigation systems, based on a background of usual routes.

Figure 3: Road network of Porto.

Routing algorithms and displaying of computed routes are also implemented using Spatial-Yap and its tabling engine. ILP systems provide the support for inference over geospatial data, such as GPS logs.

Correction of Automatic Classification of Forests based on Spatial Analysis

Another interesting project where we are using Spatial-Yap aims at the global monitoring of biodiversity change (9). Governments have set the ambitious target of reducing biodiversity loss by the year 2010, and scientists now face the challenge of accessing the progress made towards this target. The European Corine Land Cover (CLC) project[22] provides data for two different years (1990 and 2000), using 44 land-cover classes. This data is obtained from satellite images and the vectorization in the polygons of each of the 44 land-cover classes is done automatically, based on color recognition. Unfortunately, the land-cover classes of CLC are not the most appropriate to monitor biodiversity. For instance, currently CLC has only three classes for forest (broad-leaved, coniferous and mixed); therefore, an observed increase in broadleaved forest area could be due to an increase in plantation area of an exotic species, such as Eucalyptus globulus, or an increase in native broad-leaved forest, two phenomena with different implications for biodiversity. The group of biologists with whom we are working has detailed regional maps from the area of Alto Minho, in Portu-

[22]The European Corine Land Cover (CLC) project: http://terrestrial.eionet.eu.int/CLC2000

gal (see Fig. 4), also covered by CLC. These regional maps are done based on expensive and slow on-site mapping techniques and on-site identification of forest species, allowing a much higher detail on the list of classes. Our project is trying to use these detailed regional maps to derive a set of spatial logic rules that allow the detailed characterization of CLC data for biodiversity monitoring. We are using Spatial-Yap and the APRIL ILP system over data-sets created based on the intersection of the regional maps and CLC maps. The inducted rules can then be used to improve the categorization of new CLC data, allowing its use for biodiversity monitoring.

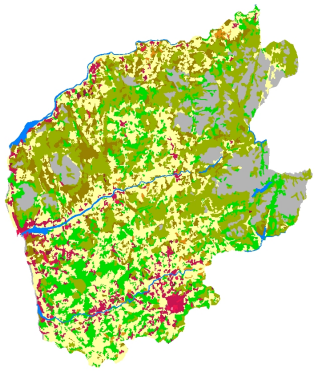


Figure 4: Alto Minho Map.

Future Work

A migration from MySQL to PostgreSQL is undergoing. Our goal is to extend the Prolog to SQL compiler in order to be able to transfer to the database system conjunctions of logic goals, that can take advantage of spatial indexing that currently is not available in Spatial-Yap.

Spatial-Yap can be downloaded from `myddas.dcc.fc.up.pt`. A users manual and several papers with deeper presentations of Spatial-Yap are also available from the same webpage.

Acknowledgments

This work has been partially supported by MYDDAS (POSC/EIA/59154/2004) and by funds granted to LIACC through the Programa de Financiamento Plurianual, Fundação para a Ciência e Tecnologia and Programa POSC. David Vaz is funded by FCT PhD grant SFRH/BD/29648/2006.

Bibliography

[1] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, 1989.

[2] V. Santos Costa, L. Damas, R. Reis, and R. Azevedo. *YAP User's Manual*. Available from `http://www.ncc.up.pt/~vsc/Yap+`.

[3] P. C. Kanellakis, G. M. Kuper, and P. Z. Revesz. Constraint query languages. *J. Comput. Syst. Sci.*, 51(1):26–52, 1995.

[4] R. Kowalski. Predicate Logic as a Programming Language. In *Information Processing*, pages 569–574. North-Holland, 1974.

[5] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1987.

[6] S. Muggleton. Inductive Logic Programming. In *Conference on Algorithmic Learning Theory*, pages 43–62. Ohmsma, 1990.

[7] T. Soares, M. Ferreira, and R. Rocha. The MYDDAS Programmer's Manual. Technical Report DCC-2005-10, Department of Computer Science, University of Porto, 2005.

[8] T. Soares, R. Rocha, and M. Ferreira. Generic Cut Actions for External Prolog Predicates. In P. V. Hentenryck, editor, *Proceedings of the 8th International Symposium on Practical Aspects of Declarative Languages, PADL'2006*, number 3819 in LNCS, pages 16–30, Charleston, South Carolina, USA, January 2006. Springer-Verlag.

[9] H. M. Pereira and H. D. Cooper. Towards the global monitoring of biodiversity change. *Trends in Ecology & Evolution*, 21(3):123–129, March 2006.

David Vaz
DCC-FC & LIACC, University of Porto
`davidvaz AT dcc.fc.up.pt`

Michel Ferreira
DCC-FC & LIACC, University of Porto
`michel AT dcc.fc.up.pt`

Case Studies

The DIVERT Project: Development of Inter-Vehicular Reliable Telematics

Hugo Conceição, Luís Damas, Michel Ferreira and João Barros

Introduction

The advent of wireless ad-hoc car-to-car (C2C) networks, i.e. groups of spatially-aware vehicles equipped with the ability to communicate over the ether and to self-organize into a collaborative mesh, opens a myriad of possibilities towards sharing and exploiting highly dynamic geospatial information. The CAR 2 CAR Communication Consortium Manifesto (1) describes several scenarios where these networks are used for purposes such as the improvement of driving safety (2), optimization of traffic efficiency or to provide information and entertainment to the driver. Examples of safety applications are *cooperative forward collision warning* (see e.g. (3)), *pre-crash sensing/warning* or *hazardous location notification*.

Regarding traffic efficiency, applications include *enhanced route guidance and navigation*, where equipped vehicles use information collected either from road infrastructures of from other vehicles about current traffic conditions to calculate optimal routes to destinations. Another application is the *green light optimal speed advisory*, where a traffic sign is able to transmit to vehicles the optimal speed to make their driving smoother and avoid stopping. Similarly, wireless communication between nearby vehicles can provide a *C2C merging assistance* to allow cars to join flowing traffic without disrupting it.

Wireless ad-hoc C2C networks will also enable applications not directly connected to safety or traffic efficiency, such as *point of interest notification* broadcasted to vehicles from local businesses or tourist attractions, or *remote diagnostics* of vehicles. Another application will be *internet access in vehicle* through the C2C network, allowing multi-hop routes to internet access points.

The deployment of a C2C network and its applications face some relevant challenges. It is clear that an inter-vehicular communication technology requires a significant distribution in the market before it can show any effect. The C2C Communication Consortium has estimated a required penetration rate of about 5% to enable traffic information propagation. A reluctant introduction may prevent potential new customers from equipping their vehicles with such communication systems. The scalability of the C2C communication system is another important issue that has to be studied. The system must work in scenarios with very small density of road traffic and in situations with a very high traffic density, which cause different technical challenges. The

development of collaborative navigation protocols is another major challenge faced by C2C networks. The goal of propagating traffic information is to allow vehicles to dynamically calculate the fastest route to their destination. Clearly, such dynamic calculation must be based on inter-vehicular collaboration, diverting routes in a global optimization perspective of the road network.

Given these challenges and the complexity in modeling the mobility behavior of large-scale distributed traffic systems (see e.g. (4)), the development of realistic simulation tools is arguably a vital element towards the success of the implementation of a C2C network. Motivated by this need, we present an open-source real-time simulation framework for moving vehicles in different road environments, that includes multiple driving states, inter-vehicle communication and sophisticated visualization. Our simulator provides the basis for a systematic approach towards quantifying the performance trade-offs between relevant metrics such as transmission radius, fraction of communicating vehicles, freshness of data, and network connectivity, thus highlighting the dynamics of cooperative navigation.

The DIVERT Simulator

From an abstract point of view, the road network can be seen as a large graph, whose topology is static and determined by geography, on top of which we have a random communications graph, whose spatial realization and connectivity pattern at each point in time is determined both by the position of the vehicles moving on the road network and by the transmission aspects of the wireless interface they use to communicate. To obtain a realistic road graph model, we may resort to increasingly available geospatial information, whereas the wireless transmission characteristics have been the object of intense study yielding useful random models with varying complexities (see e.g. (5)). Given these two aspects, we structured our simulation prototype (named DIVERT - Development of Inter-VEhicular Reliable Telematics) in two layers: a traffic simulation layer based on the road network graph; and a wireless telematics layer, based on the random communications graph. We next describe these layers.

Traffic Layer

The geospatial information over which the traffic simulation layer operates is conveyed to DIVERT using widely used formats, such as shapefiles, which describe the geometry and connectivity of the road network. DIVERT includes a sophisticated user interface which allows editing the basic road segments enriching them with low-level information describing traffic entities. A screenshot of this interface is shown in Fig. 1.

Currently, DIVERT has been setup using geospatial information of the city of Porto, the second largest in Portugal. Its road network covers an area of 62 square kilometers, with 1941 streets summing up to 965 kilometers of total length.

In DIVERT we model two types of vehicles: vehicles which circulate and communicate, called *sensors*; and vehicles which just circulate. Within each type of vehicle, DIVERT further distinguishes in normal and large-sized vehicles, associating appropriate mobility patterns to each. These mobility patterns are also individually influenced by random initialization of attributes such as acceleration, braking, aggressiveness and risk tolerance. Sensors add an attribute of wireless transmission range.

DIVERT uses the following layers of geospatial information about the road network:

1. Two simple layers of the road central axles, representing, through polylines, the geometry of intersection free road segments, and their topological connectivity. These layers can be given to DIVERT as shapefiles. A copy of these layers is present in every sensor, and is used for positioning of GPS readings and for the collaborative propagation of mobility conditions on road segments.
2. Low-level layers describing in detail the road network of Porto, including information of road segment lanes, lane-level connectivity, intersection visibility, traffic lights location, traffic lights inter-relationships, speed limits on segments, and parking. These layers must be edited through the DIVERT interface and are used by the traffic simulator.

A raster layer from satellite images of the simulation area further improves the visualization of traffic in our prototype, which is currently based on 2D data. Work is undergoing on a 3D model, which will not only allow more realistic accelerations parametrized by steepness, but also enable an improved modeling of wireless transmission ranges, which accounts for fading, reflection and shadowing effects based on a 3D layer for buildings.

Regarding vehicle routes, DIVERT currently uses an hybrid model of pre-defined routes and randomly

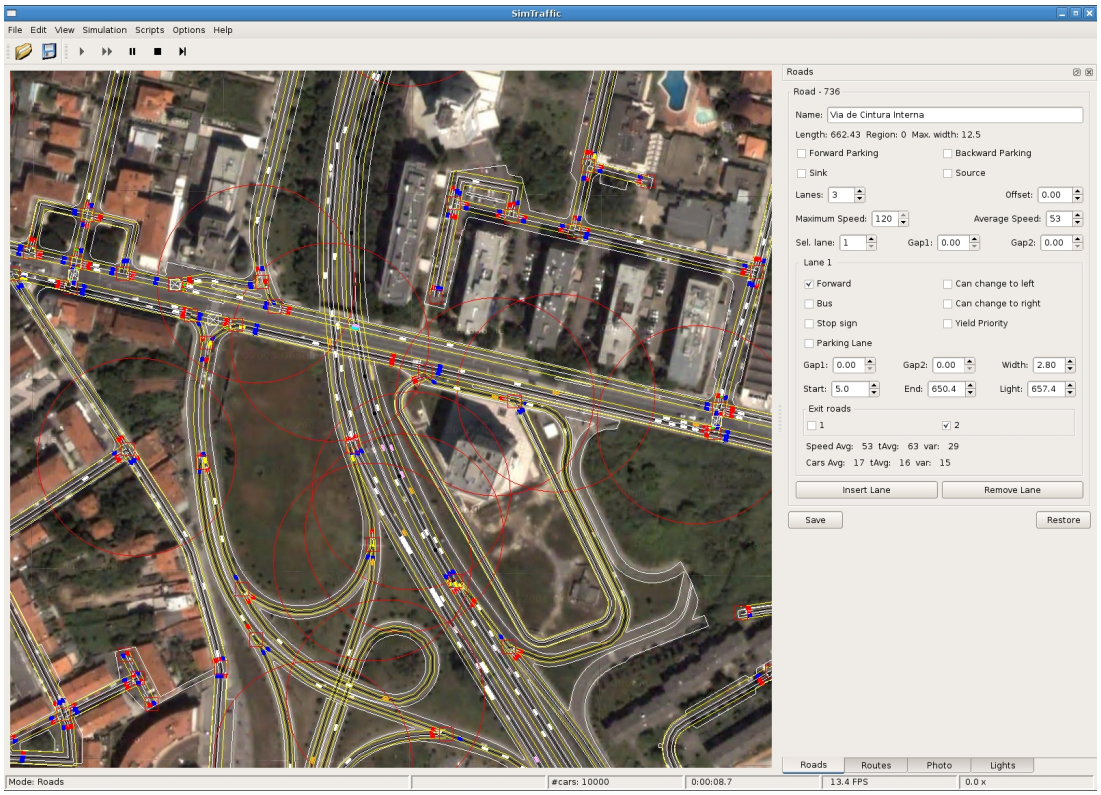


Figure 1: The DIVERT road editing interface

generated routes. For randomly generated routes, our system arbitrarily selects an origin and a destination and calculates the route based on a shortest-path algorithm, either parameterized by distance or by time. Shortest-path based on time uses not only the speed limits of segments, but mainly the dynamic calibration of average mobility derived from previous simulation results. Pre-defined routes have an associated frequency and have been carefully chosen to approximate the simulation to our perception of traffic distribution in our current work case, the city of Porto. Figure 2 shows the DIVERT interface for setting up a pre-defined route.

Traffic simulation is parameterized by the number of vehicles and the percentage of these vehicles which are sensors. Simulation is initialized by randomly placing each vehicle in an arbitrary point of its route. Vehicles which arrive at their destination are removed. New vehicles also show up during the simulation, either from entry points in the map, or from parking lanes of segments, as DIVERT tries to maintain the targeted number of vehicles for the simulation.

It should be noted that the simultaneous micro-simulation of thousands of vehicles, with the degree of sophistication offered by DIVERT, poses major challenges in term of optimization of algorithms and efficiency of data-structures. In particular, DIVERT implementation is multi-threaded, exploring multi-core architectures of current processors. A geographic partitioning of the simulation region is performed, allowing each partition to be independently simulated in a thread.

Wireless Telematics Layer

In order to capture the inter-vehicle communication aspects it is necessary to define the level of abstraction with respect to the physical communication channel and the protocol architecture. At the current preliminary stage, we opted for a very simple model, in which vehicles communicate with each other if their distance is below a certain threshold, determined by the transmission radius. The resulting random geometric graph is widely accepted as a simple yet reasonable first-order approximation of the connectivity pattern attained by a mobile ad-hoc network (6). A more elaborate approach would be to consider path loss, multi-path and shadowing effects, however this would incur in a high penalty

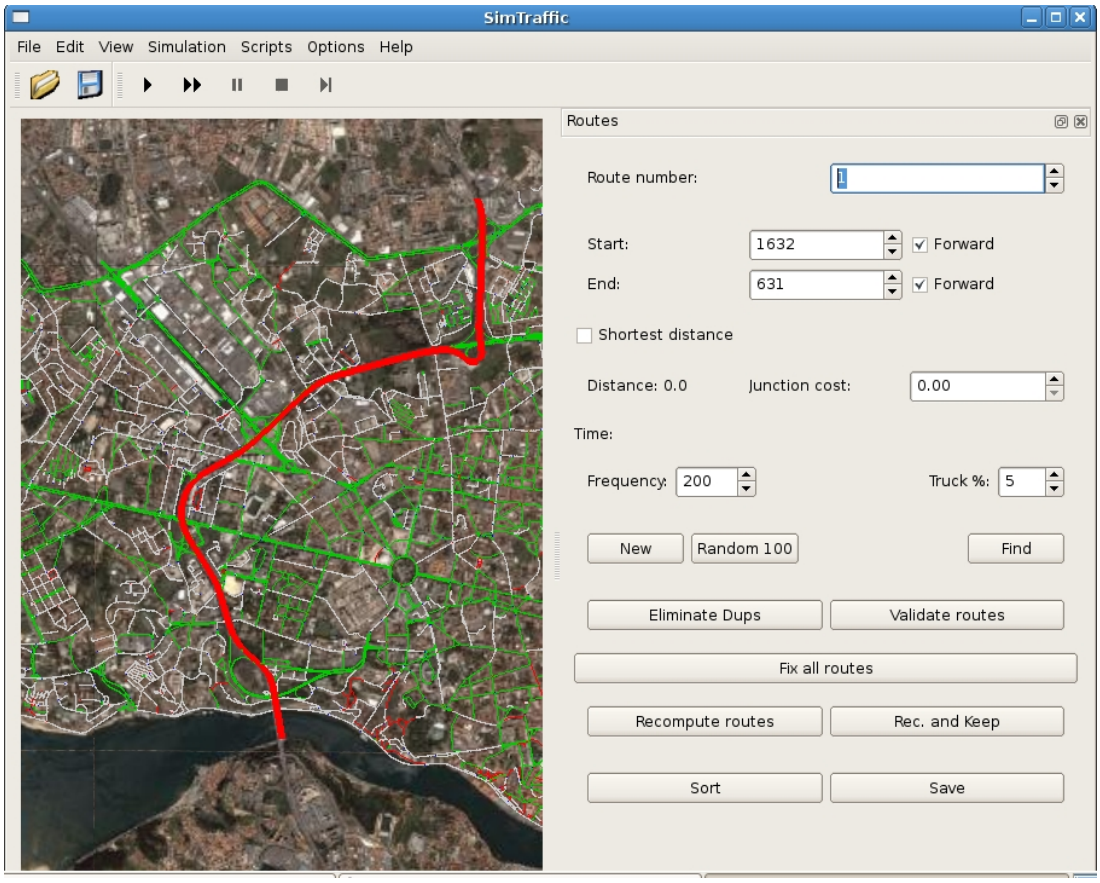


Figure 2: Setting up pre-defined routes in DIVERT

in terms of simulation complexity. Another alternative would be to consider collisions and packet losses over the wireless medium. We are currently considering the possibility of integrating these aspects in our simulation in order to obtain a richer connectivity profile.

The wireless telematics layer simulates the communication between vehicles. Several inter-vehicular protocols can be implemented in this layer, such as protocols for safety applications or for traffic conditions propagation and collaborative navigation. The implementation of this wireless telematics layer is supported by the traffic simulation layer, which acts as a global positioning server, emulating a GPS receiver in each of the sensors. The interface between the two layers is thus done through GPS-like sentences, where the traffic simulator generates the position of each sensor, in terms of latitude and longitude, and its velocity vector, together with a global timestamp which provides the time-synchronization among the inter-vehicle data exchange. It is a task of the wireless telematics layer to calculate the vehicles in the (parameterized) transmission range of a given sensor, to implement the message exchange protocols between vehicles, and to trigger actions based on the information collected, such as a route modification. The architecture of the two layers in DIVERT allows that a triggered action such as route modification can be conveyed back to the traffic simulation layer, affecting the behaviour of the vehicle.

Currently, the protocols we have implemented using DIVERT aim at the collaborative propagation of mobility information about road segments. Each sensor stores a common data structure, where the pair $(AverageSpeed, Freshness)$ describes the average speed attained by sensors traversing each of the road segments, together with a quantification of the timeliness of average speed estimate. These pairs are updated either by each sensor based on the road it is traversing and its GPS information, or by aggregated information collected from the wireless communications broadcasted by other sensors. Typically, sensors only transmit information about road segments for which the freshness value is above a predefined

threshold. DIVERT is able to simulate the propagation of this mobility information using different wireless transmission radii. Our results show that there exists a critical value for the transmission radius after which the dissemination of traffic information is sufficient for a large number of vehicles to be able to compute a comprehensive and accurate congestion map. This observation is strikingly related to the physical phenomenon of percolation, which is well known to govern the connectivity of large classes of wireless networks (7). Once the transmission radius is above the critical threshold, the graph representing wireless connectivity has a giant component on which traffic information flows very fast and over long distances.

Future Developments

DIVERT is under constant development. Currently the implementation lists 50000 lines of C++ code, including the graphical interface and visualization component. The DIVERT interface also allows launching simple programs written in Python, which are very useful to make the simulator produce several types of reports, generate videos of a simulation or help in the edition of maps. A redefinition of the simulator architecture in several independent modules is undergoing. Our goal is to have a larger number of independent modules which will make it simpler for a large community of users to modify the simulator to their specific needs.

A particular effort is being put on the development of a specification language that will allow making easier the setting up of DIVERT with different geospatial data. The geographic layers underpinning the simulation are already based on open standards, but there are still a number of areas where the lack of automation constitutes an obstacle to the wide use of DIVERT. In particular, the realistic calibration of routes and their frequency is a crucial problem. We are trying to approach such automation through the analysis and processing of cellular phone logs, a technique known as *floating car data*, which in urban environments are able to provide high-precision descriptions of travels in the road network. The automation of the construction of realistic origin/destination matrices, together with geospatial data defining the road network based on widely used standards, would provide the necessary data to test DIVERT in different scenarios.

We will continue designing, implementing and testing different protocols for inter-vehicular communications through DIVERT. Our focus will continue to center in traffic efficiency, where we have been able to find challenging problems related to collaborative optimization of traffic flow. We hope to see alternative protocols developed worldwide using the DIVERT framework, in all areas of C2C communication.

For those who missed the FOSS4G2007 DIVERT demonstration, a video of a simulation is available online.[23]

Acknowledgments

We would like to thank the Porto City Council for providing us with the maps of the road network. This work has been partially supported by the Portuguese Foundation for Science and Technology (FCT) under project MYDDAS (POSC/EIA/59154/2004) and by funds granted to LIACC and IT through the Programa de Financiamento Plurianual and POSC.

Bibliography

[1] CAR 2 CAR Communication Consortium Manifesto. Version 1.1 http://www.car-2-car.org/, 2007.

[2] C. L. Robinson, L. Caminiti, D. Caveney, and K. Laberteaux. Efficient coordination and transmission of data for cooperative vehicular safety applications. In *VANET '06: Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, pages 10–19, New York, NY, USA, 2006. ACM Press.

[3] X. Yang, J. Liu, F. Zhao, and N. H. Vaidya. A vehicle-to-vehicle communication protocol for cooperative collision warning. In *MobiQuitous*, pages 114–123. IEEE Computer Society, 2004.

[23] FOSS4G2007 DIVERT demonstration: http://myddas.dcc.fc.up.pt/divert/

[4] A. K. Saha and D. B. Johnson. Modeling mobility for vehicular ad-hoc networks. In *VANET '04: Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, pages 91–92, New York, NY, USA, 2004. ACM Press.

[5] P. Gupta and P. Kumar. The capacity of wireless networks. *Information Theory, IEEE Transactions on*, 46(2):388–404, 2000.

[6] F. Xue and P. Kumar. The Number of Neighbors Needed for Connectivity of Wireless Networks. *Wireless Networks*, 10(2):169–181, 2004.

[7] L. Booth, J. Bruck, M. Franceschetti, and R. Meester. Covering algorithms, continuum percolation and the geometry of wireless networks. *Ann. Appl. Probab*, 13(2):722–741, 2003.

Hugo Conceição
DCC-FC & LIACC, University of Porto
hc AT dcc.fc.up.pt

Luís Damas
DCC-FC & LIACC, University of Porto
luis AT dcc.fc.up.pt

Michel Ferreira
DCC-FC & LIACC, University of Porto
michel AT dcc.fc.up.pt

João Barros
DCC/FC & IT - University of Porto
barros AT dcc.fc.up.pt

GRASS GIS and Modelling of Natural Hazards

An Integrated Approach for Debris Flow Simulation — First results of an application in the Central Andes

Martin Mergili and Wolfgang Fellin

Background

Debris flows are rapid mass movements of water and debris, constituting a considerable hazard when interfering with people, buildings, or infrastructure. They are often triggered by heavy or prolonged rainfall or by extreme snow melt. Mobilization of the material occurs due to translational or rotational failure of saturated or undercut slopes, or by detachment due to surface runoff or the debris flow itself. Various models do exist for simulating sub-processes included into debris flows, for example for detachment (*r.sim.sediment* within the GRASS GIS environment), for soil hydrology and slope stability (14), or for debris flow runout (9; 7). More integrated GIS-based approaches as attempted for example by (1) or (11) are scarce. Such approaches would be valuable for a quick assessment of hydrological thresholds for potential debris flow hazard regarding specified features at risk. This paper describes and discusses the development of such a model as GRASS GIS raster module. The model is designed for small catchments (few square kms) and is tested at the moment with seven study areas along the international road corridor from Mendoza (Western Argentina) to Central Chile, crossing the highest section of the Andes (figure 1). The preliminary results for the study area *Guido A* are presented.

Model

Implementation and model design

The simulation model is implemented as a GRASS GIS raster module called *r.debrisflow*, based on the C programming language. Data management is facilitated using shell scripts. The model is in an intermediate stage of development right now, with major technical and methodical enhancements prospected for the near future. Additionally, a GUI for data management shall be created. By now, the latest development version can be downloaded from the homepage of the first author. *r.debrisflow* constitutes of a framework of a number of sub-modules described in more detail below, the general model design is illustrated in figure 2. The sub-modules can be combined in two different ways, depending on the availability of input information:

Simulation mode 1: The entire hydrological, stability, detachment and runout modelling is executed for a defined number of time steps during a rainfall or snowmelt event, requiring an extensive set of information as input, including

Figure 1: Study areas. The preliminary results for Guido A are presented.

meteorological data, an elevation model, soil mechanical and hydrological parameters and surface hydrological characteristics (including land cover).

Simulation mode 2: The zones of debris flow initiation are defined manually (e.g. from mapping in the field or from orthophotos), and only runout is computed. The advantage of this mode is that it requires much less input than the others, but, on the other hand, it is not suitable for predicting future events.

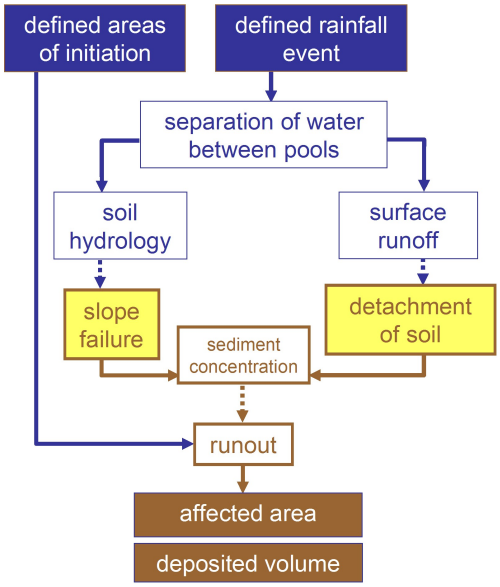


Figure 2: General model design.

Water input

Rainfall is read from the input file and added to the system by increasing the surface water table of each cell, reduced by interception. If a snow cover exists, snowmelt is computed for each cell with a user defined degree-day-factor and added to the surface water table.

Soil hydrology and slope stability

For this sub-module, a three-dimensional raster approach is used, down to the depth of bedrock (if known), or to a user-defined maximum soil depth. The soil is assumed to be homogeneous over its entire depth regarding its physical, hydrological and mechanical properties. Vertical flow between cells is computed with the Darcy-Equation. If the water content of a cell exceeds 90 % of the maximum content, groundwater flow is assumed to be parallel to the slope and it is tested whether the cell is stable or not, using an infinite slope stability approach (14). For each pixel, the bottom of the deepest cell with a factor of safety lower than 1 is considered as failure plane (figure 3). It has to be pointed out that this approach constitutes a rough approximation to the reality with the character of a worst-case assumption: the stabilizing role of vertical water movement is neglected, and the destabilizing role of the assumed slope-parallel component is fully included in case of saturation. In the real world, both components are combined, resulting in more stable conditions than in the model.

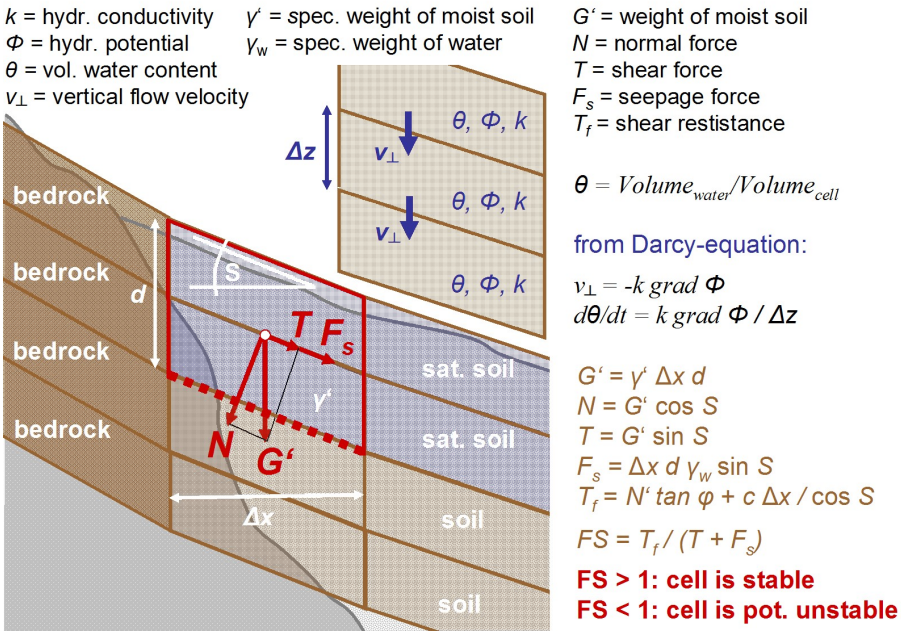


Figure 3: Subsurface hydrology and slope stability model.

Surface runoff and detachment

All the water which is not infiltrated into the soil is added to the surface water table of the corresponding cell. Flow velocity from one cell to the other is computed using the Manning equation. If no flow channel is defined for any downward cell, flow takes place to all downward cells, weighted for slope angle. If a channel is defined, the entire flow moves there. Transport capacity is computed using the (15) equation, which it is supposed to suit best for the conditions in the study areas (6). Rates of detachment are derived from the transport capacity. The model design is illustrated in figure 4.

Debris flow runout and deposition

Debris flows are here understood in a strict sense, with a non-hydraulic flow regime and excluding heavily sediment-loaded water discharge. For this reason, the sediment concentration is computed for the mobilized soil volume of each cell:

1. If the sediment concentration exceeds a threshold value (6), the entire mobilized volume is considered to develop into a debris flow. If mobilization occurred due to slope failure, the cell is marked and runout of all the failed cells is computed at the end of the event. If mobilization occurred due to detachment, runout is calculated immediately at the end of the time step. This is a first approximation as the empirical runout models used (compare below) neglect the time required for runout.
2. If the sediment concentration is too low for the development of a debris flow, the material is removed with surface runoff and either deposited downslope or removed from the system, following the (15) equation (compare figure 4). Though deposition from surface runoff does not fall into the concept of debris flow, it can provide valuable complementary information and is therefore regarded, too.

Debris flow runout itself can be simulated using physically-based models (9; 7), but they are complex and hard to be integrated into a GIS environment. Therefore it was decided to use a combination of empirical approaches first for estimating the runout distance and the distribution of the deposited volume (figure 5). (10) developed an approach enabling the distinction between scouring and deposition areas, using threshold slope angles and the ratio between vertical distance of scouring and horizontal distance of deposition. (2) and (8) used mobilized volume, angles, and runout distances for estimating the reach of the debris flow. These approaches, however, have the disadvantage that they don't allow distinguish-

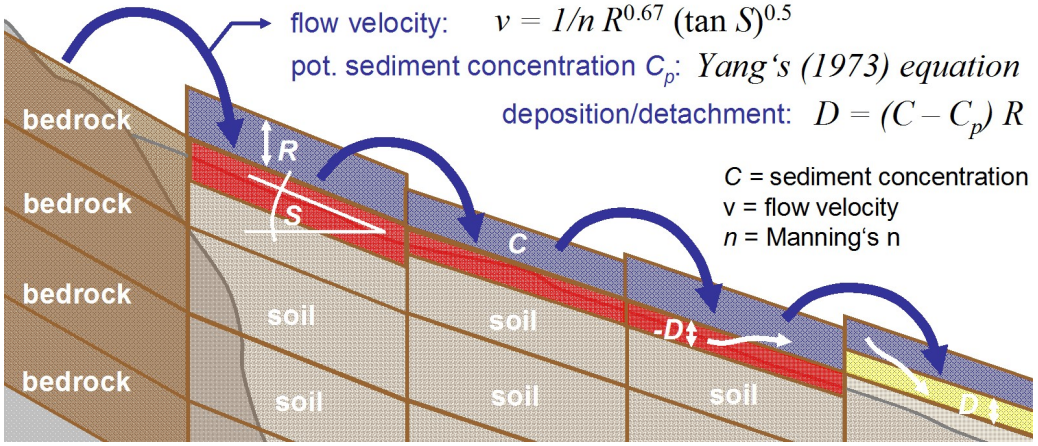


Figure 4: Surface runoff and detachment model.

ing between areas of scouring and deposition, and therefore the distribution of the deposited material. The approach was included into the model as follows:

1. The Vandre approach was applied with user-defined parameters for estimating distributed scoured and deposited volumes.
2. The Corominas et al. and the Rickenmann approach were applied independently and then combined to an index.

The debris flow is routed downwards separately for each unstable cell, following a random walk (3) weighted for slope angle and the existence of a defined channel, until the stop criteria for all of the three approaches is fulfilled. Though each cell is treated separately, the mobilized volume required for runout distance is calculated for each patch of debris flow initiation. In the area of scouring, the entire saturated soil column is considered to be removed, but never exceeding the depth of initiation. The initiated and scoured volumes are considered to distribute over the area of deposition as wedge shape rising towards the front.

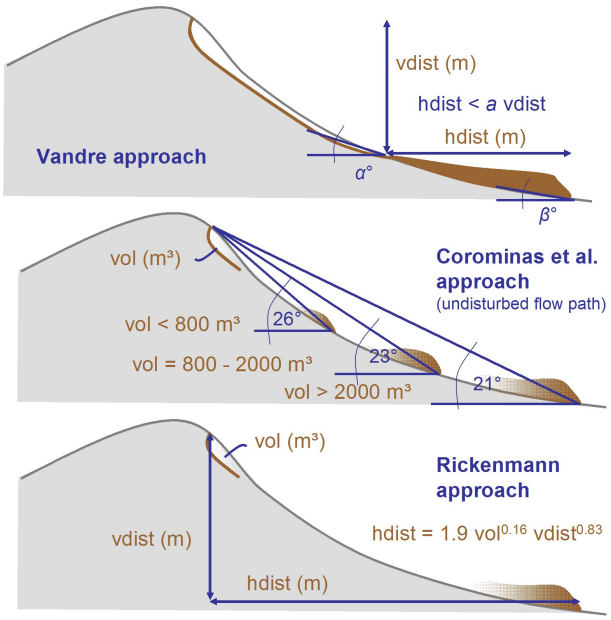


Figure 5: Runout models.

First results

The model was tested within the study area Guido A (compare figure 1). Mainly consisting of granite residuals, the soils of the catchment (area: 2 km^2) are relatively homogeneous. Therefore it was decided to use one single set of soil parameters for the entire area:

texture	ρ_d kg/m^3	c_s N/m^2	φ deg.	Θ_s	k_f cm/h
Sand	1850	0	40.0	0.43	29.7

ρ_d is the dried bulk density of the soil, c_s is soil cohesion, φ stands for the angle of internal friction, Θ_s is the maximum (saturated) water content, and k_f is the saturated hydraulic conductivity.

The figures 6 and 7 illustrate the mapped areas of debris flow initiation in the study area Guido A, and the patterns of surface change due to a debris flow event, based on the mapped areas of initiation and the computed patterns of scouring and deposition (simulation mode 2). The white line crossing the right part of the maps represents the international road, roughly coinciding with the distal part of the observed debris flow depositions. The figures 8 to 11 show some of the simulation results for a hypothetical 100 mm rainfall event, corresponding to the maximum daily sum ever recorded at the nearby meteorological station, and therefore constituting a worst-case assumption (simulation mode 1). All maps show plausible patterns when compared to field observations. The areas of debris flow initiation and deposition are located correctly, but are overestimated compared to the patterns observed in the field (what is not surprising for a worst-case assumption). The calculated sediment volumes deposited on the international road are within the same magnitude as those reported by the road authorities. When simulating the impacts of smaller rainfall events, the model results correspond well to the findings of (5) that debris flows in the Mendoza valley usually occur at daily rainfall sums exceeding 6.6 to 12.9 mm.

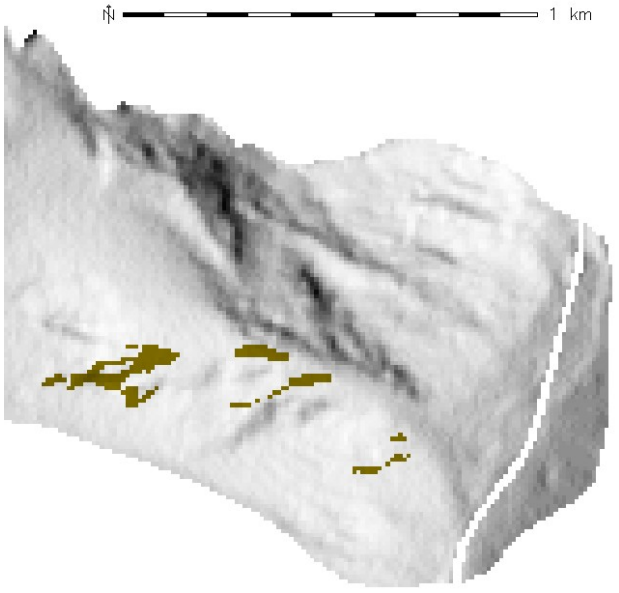


Figure 6: Mapped areas of clearly identifiable previous debris flow initiation, depth of initiation assumed as 0.75 m according to field evidence.

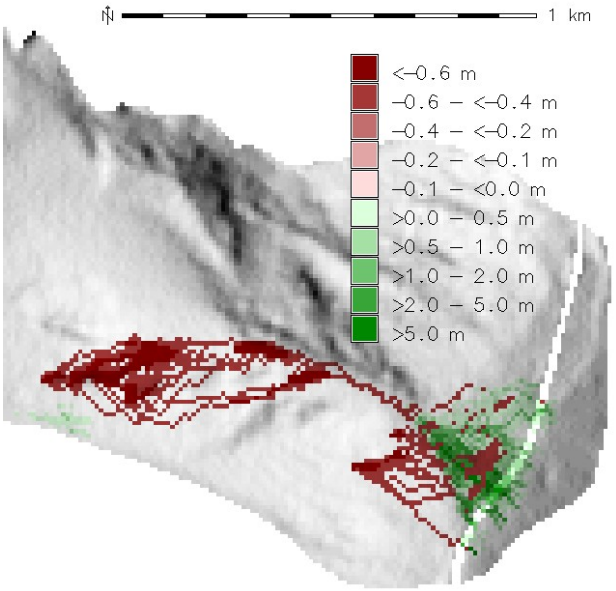


Figure 7: Simulated change of terrain height due to debris flow, using mapped areas of initiation.

Discussion and Preview

Though the preliminary results for Guido A appear plausible, the simulation model still shows a number of shortcomings that have to be reworked.

1. Infiltration of water into the soil is not yet modelled in a satisfactory way, so that the approach will have to be refined (Green-Ampt model). Slope-parallel groundwater flow will be included, too, for enabling a closer approximation to the reality of soil hydrology.
2. The slope stability model as applied at the moment is only valid for plane, infinite, cohesionless slopes. For very shallow failures, this assumption is sufficiently close to reality, but for more deep-seated rotational failures, it is unsatisfactory and slope curvature has to be taken into account. (12) and (13) could serve as examples for such an approach.
3. The empirical approaches for debris flow runout shall be complemented by the implementation of a physically-based runout model according to (9) and (7), or at least of an interface to a non GIS-based runout model.
4. An interesting extension would be to introduce some probabilistic elements into the slope stability model (regarding the SINMAP model as an example) and into the distinction rules between debris flow and other types of movements.

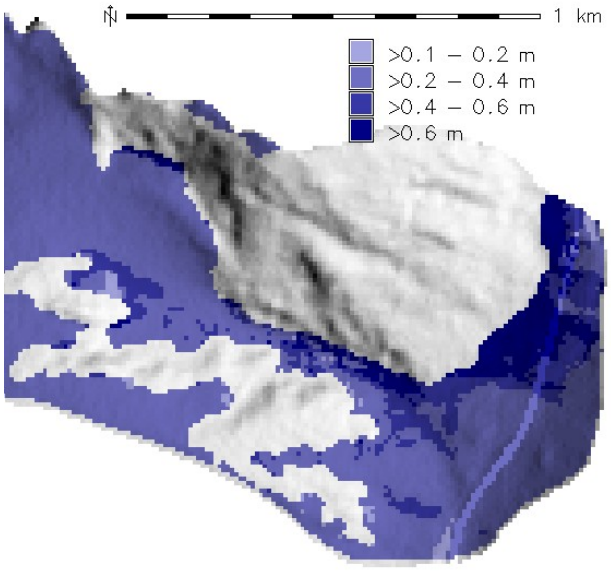


Figure 8: Maximum depth of saturation computed for 100 mm rainfall event.

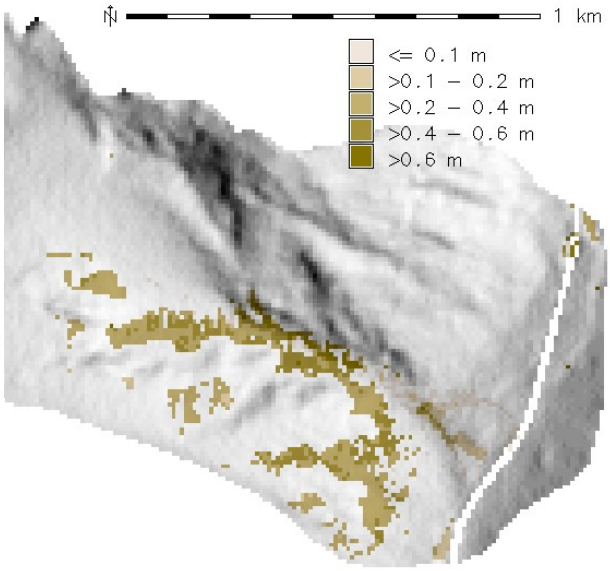


Figure 9: Simulated areas of potential debris flow initiation computed for 100 mm rainfall event.

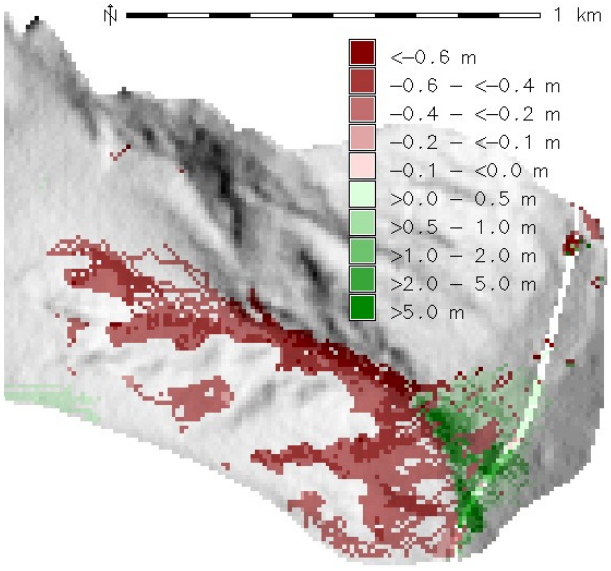


Figure 10: Simulated change of terrain height due to debris flow caused by 100 mm rainfall event.

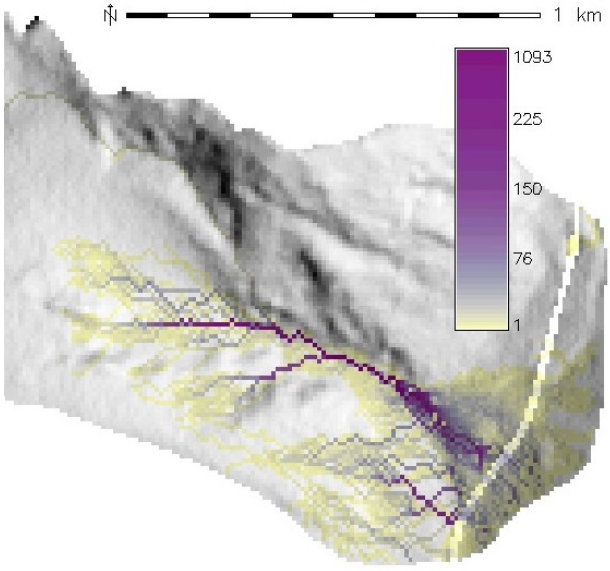


Figure 11: Debris flow index for 100 mm rainfall event, denoting number of cells the mobilized material of which hits the pixel.

5. Finally, the model has to be applied to the remaining study areas (compare figure 1), and the results have to be tested carefully against the field observations and the validation data (reports about volumes of material deposited on the international road).

With the mentioned optimizations, r.debrisflow shall be a valuable tool for evaluating the potential magnitude of debris flows as a response to defined rainfall or snow melt events, including the possibility to determine meteorological thresholds for debris flow hazard. However, it has to be pointed out that all the results only denote potential occurrences with the character of worst case scenarios or probabilities - it will probably never be possible to predict the actual response of a slope to a meteorological event in the real world, as nature is too complex to be fully understood in all details.

Acknowledgements

The project is funded by the *Doktoratsstipendium aus der Nachwuchsförderung der LFU* and by grants from the Office for International Relations and the Faculties for Natural Sciences, University of Innsbruck, as well as from the Austrian Academy of Sciences and the Federal Government of Upper Austria. Thanks to Hans Stoetter, Axel Borsdorf, Clemens Geitner and Stella Moreiras for their support.

Bibliography

[1] A. Burton, J.C. Bathurst (1998) Physically based modelling of shallow landslide sediment yield at a catchment scale. *Environmental Geology* 35,2-3: 89-99.

[2] J. Corominas, R. Copons, J.M. Vilaplana, J. Altamir, J. Amigó (2003) Integrated Landslide Susceptibility Analysis and Hazard Assessment in the Principality of Andorra. *Natural Hazards* 30: 421-435.

[3] B. D. Hughes (1995) Random walks and random environments: Volume 1: Random Walks. Oxford University Press. 652 pp.

[4] X. Li (2007) Finite element analysis of slope stability using a nonlinear failure criterion. *Computers and Geotechnics* 34: 127-136.

[5] S.M. Moreiras (2005) Climatic effect of ENSO associated with landslide occurrence in the Central Andes, Mendoza Province, Argentina. *Landslides* 2: 53-59.

[6] J.S. O'Brien (2003) FLO-2D Users' Manual Version 2003.06, July 2003. FLO-2D Software Inc., Nutrioso, Arizona, USA. 232 pp.

[7] S.P. Pudasaini, K. Hutter (2007) Avalanche Dynamics: Dynamics of rapid flows of dense granular avalanches. Springer, Berlin Heidelberg. 602 pp.

[8] D. Rickenmann (1999) Empirical Relationships for Debris Flows. *Natural Hazards* 19: 47-77.

[9] S.B. Savage, K. Hutter (1989) The motion of a finite mass of granular material down a rough incline. *Journal of Fluid Mechanics* 199: 177-215.

[10] B.C. Vandre (1985) Rudd Creek debris flow. In: D.S. Bowles (ed): Delineation of landslide, flash flood, and debris flow hazards in Utah. Utah Water Research Laboratory, Utah State University, Logan, Utah, 117-131.

[11] V. Wichmann (2006) Modellierung geomorphologischer Prozesse in einem alpinen Einzugsgebiet - Abgrenzung und Klassifizierung der Wirkungsräume von Sturzprozessen und Muren mit einem GIS. Eichstaeter geographische Arbeiten 15: 231 pp. In German.

[12] P.L. Wilkinson, M.G. Anderson, D.M. Lloyd (2002) An integrated hydrological model for rain-induced landslide prediction. *Earth Surface Processes and Landforms* 27: 1285-1297.

[13] M. Xie, T. Esaki, G. Zhou, Y. Mitani (2003) Three-dimensional stability evaluation of landslides and a sliding process simulation using a new geographic information systems component. *Environmental Geology* 43: 503-512.

[14] M. Xie, T. Esaki, M. Cai (2004) A time-space based approach for mapping rainfall-induced shallow landslide hazard. *Environmental Geology* 46: 840-850.

[15] C.T. Yang (1973) Incipient motion and sediment transport. *Journal of the Hydraulics Division. Proceedings of the American Society of Civil Engineers* 99: 1679-1703.

Martin Mergili
University of Innsbruck, Institute of Geography
and Austrian Academy of Sciences, Mountain Research
www.uibk.ac.at/geographie/personal/mergili
martin.mergili AT uibk.ac.at

Wolfgang Fellin
University of Innsbruck, Institute of Infrastructure
www.uibk.ac.at/geotechnik/staff/fellin-en
wolfgang.fellin AT uibk.ac.at

A Spatial Database to Integrate Information of the Rondonia Natural Resource Management Project

Luis Fernando Bueno, Luiz Gilberto Dall'Igna, Marcelo Vitor Amaral Campos, Thiago de Lima Martarole, Diego Gomes Ferreira

Keywords: *Spatial Database, ZSEE-RO, PLANAFLORO, Data Integration*

Abstract

This paper presents the construction of a spatial database for the integration information from Social-Economic Ecological Zoning of Rondonia State (ZSEE/RO). The project's objective consists of modeling a new database for ZSEE/RO, making it possible to access the data and information for use in varied projects. So all the evaluation was performed based on the quantity of ZSEE-RO data and information. File conversions were carried out, aimed at making the files accessible in standard formats. Later the consistency of converted data was verified, to insure quality control at different scales. Some applications had to be reconfigured and installed which were developed exclusively for the ZSEE/RO. The main results so far relate to the recognition of ZSEE/RO data quantity and establishment of the necessary requirements for accessing the original data. Diverse files have been converted to universal formats. Moreover, the problems with regard to the data quality which have been detected include non geoencoded data, inconsistencies at different scales, and inconsistency in printed matter versus stored digital files.

Introduction

This article covers spatial database construction for the integration of Social-Economic Ecological Zoning of Rondonia State (ZSEE/RO), which is being developed in partnership with Secretariat of Planning and General Coordination of the State of Rondonia. The ZSEE/RO is originally the Rondonia Natural Resource Management - PLANAFLORO.

The general objective of the project consists of modeling a new database of the ZSEE/RO, making it possible for various project to access the data. Specifically, it is intended with the elaboration of this work: to evaluate the content of diverse source databases for ZSEE/RO, to shape and construct a spatial database for storage of data and information, and to convert the data to a new internet based format.

The PlanaFloro Project

Through the State Decree No. 3782 of 14 June of 1988 the Social-Economic Ecological Zoning of Rondonia State - ZSEE/RO - was established, dividing the State into zones of ambient protection and zones for farming and agroforestry activities (4). Rondonia was the first Brazilian state to have policies of preservation of nature with the Rondonia Natural Resource Management Project - PLANAFLORO, directed to the question and of granting land for the aboriginal peoples, executed between 1992 and 1999 growing into the Social-Economic Ecological Zoning of Rondonia state.

As one of the components of PLANAFLORO the second approach of the ZSEE/RO supported a field survey at 1:250,000 scale, elaborating diverse fields of the knowledge such as: vegetation, geology, pedology, geomorphology, fauna, land use and occupation etc. The partnership responsible for execution of this mission consisted of distinct teams, one for each area of the knowledge. Consequently, The thematic data had been stored using different solutions in accordance with the necessities of the diverse teams. Its implementation was an ArcINFO geographic database, however it was very poor in attributes (Dall'Igna, 2005). The completed thematic studies became the basis for diagnosis of the State of Rondônia and, using its results, Approach of Ecological the Partner-Economic Zoning of the State was elaborated.

Initial Procedures

To define the work methodology the great volume of data concerning the ZSEE/RO was taken into

account, Original data were available in heterogeneous formats, many of them unknown to the technician of the state agencies and potential researchers. Technological obsolescence was also considered, even though PLANAFLORO was initiated in 1992. Thus the project proceeded to evaluate and collect data and information for ZSEE-RO, composing archives in digital media, customizing software for PLANAFLORO and producing printed material as maps, letters and reports. The principal files formats found included:

- ADF: ArcINFO 7.x Coverage files. In this format were themes covered by ZSEE, vector and descriptive data, airport points, curve-level, law protected areas, etc.
- LAN: Raster files. Landsat satellite images, with a 30 meters resolution, covering the entire state of Rondonia.
- GRA: Plot files. Maps generated by ZSEE for printing.
- RTL: Plot files. All themes covered by ZSEE as hydrology, geology, hydrogeology, geomorphology, vegetation, among others, and all maps prepared in scales ranging from 1:250.000 to 1:1.000.000

In addition to the formats listed above reports, pictures and spreadsheets in .DOC .XLS .TIFF .JPG and .EPS were identified.

Conversions of archives has been carried out, aiming to become accessible and standardized, converting them from proprietary files which need specific software, into universally accessible files for diverse technologies. The files that were in .ADF vectorial format were converted to the ESRI Shapefile format. The satellite images .LAN format were converted to GeoTiff and the plot files .RTL and .GRA were converted to .PDF format. The reports and pictures originally in .DOC, .TIF, .JPG and .EPS were also converted to the .PDF. All spreadsheets, database and some TXT format files were uploaded to tables on PostgreSQL. Some applications adopted and developed exclusively for the ZSEE/RO, have been reconfigured and installed. In the future, its functions will be tested to determine if they will be used, substituted or brought up to date.

Also TerraView, ArcView, ArcInfo and IDRISI are software being used. The geographic database of the ZSEE/RO was structured with ArcInfo, extensions were developed for ArcView for use in visualizations and consultations. The pedology team used IDRISI in its analyses giving the possibility of working with the data of SIGTERON as well as with the ArcInfo and ArcView.

TerraView was chosen as an alternative, being free software capable of importing shapefiles and images in GeoTIFF format, and universal formats for vector and raster files respectively. The permission for its redistribution and/or modifications is under the terms of the GNU General Public License (GPL), as published by the Free Foundation Software. TerraView is an application based on the geo library TerraLib. It manipulates vector and raster data, both stored in relational or geo-relational DBMS on the market such as ACCESS, PostgreSQL, MySQL and Oracle (7). Figure 1 demonstrates the TerraView interface.

Later the consistency of the converted data was verified, with the intention to check the quality of the information at different scales. The procedures for checking consistency of data involved georeferencing available data points, the overlap of raster files among themselves, and the verification of the geocoded data, with the links between the geometry of objects and their descriptive attribute.

ZSEE/RO Spatial Database

An infrastructure for storage and distribution of spatial data was implemented, based on free software: PostgreSQL (5), with the PostGIS extension (6), the web map servers MapServer and GeoServer (2) as well as the catalog server GeoNetwork (3).

Vector layers were converted and stored into a PostgreSQL database through the functionality provided by the spatial extension PostGIS. The descriptive data of spatial objects, contained in heterogeneous format files, such as database files in format .DBF and electronic spread sheets, were converted for PostgreSQL tables and properly linked with its geometric representation. The vector data stored included information on various topics covered by the project ZSEE/RO: geology, soils, land use and occupation, socio-economics, transmission lines, flora, fauna, climatology, and so on. Furthermore, the data refer to basically raster images from the Landsat satellite with a resolution of 30 meters and cover the entire state of Rondonia. Furthermore, descriptive data of spatial objects, contained in heterogeneous format files, such as database files in format .DBF and electronic spread sheets, are being converted to PostgreSQL tables and properly linked with the layer's geometry.

All the thematic documentation of the ZSEE/RO was organized and stored in the GeoNetwork Open-

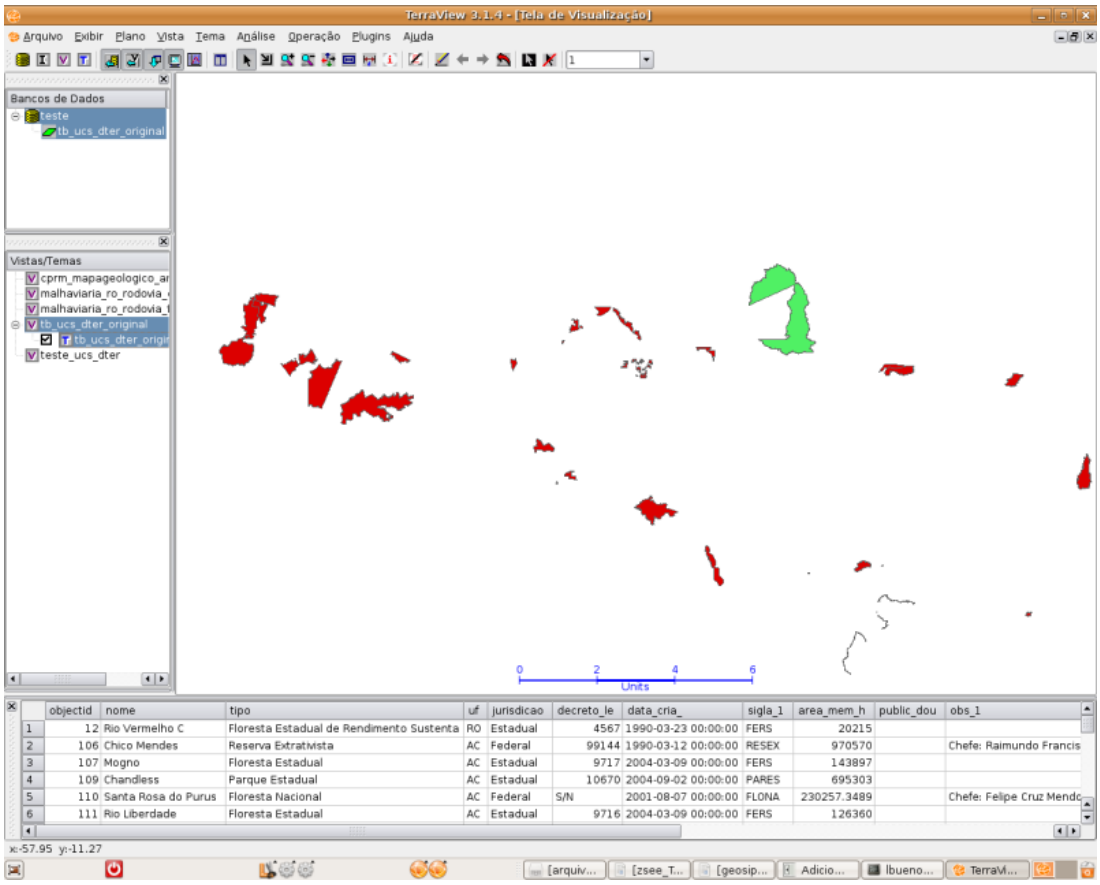


Figure 1: The Terraview interface

source. The functionality of a literal search in GeoNetwork allows easy access to a great amount of information. For each subject encompassed by ZSEE/RO, a corresponding metadata was created, where all the documentation related with the subject is posted through the resources of distribution of the GeoNetwork.

The visualization tool for spatial data chosen was I3Geo. I3Geo (Integrated Interface for Internet of Geoprocessing tools) is an application developed for access and analysis of geographic data. Based on free software, especially MapServer, it uses as its platform standard internet browsers, Internet Explorer and Firefox. I3Geo is licensed under GPL and can be used and incorporated by any interested institution with no costs. Adopting international standards of interoperability, I3Geo incorporates functions that facilitate the remote access of data, allowing for the establishment of cooperative networks. Operations that normally are found only in programs for personal computers, which operate in local installations, are available in I3Geo, such as generation of graphs, tabular data analysis, spatial operations, etc. (8).

Results

The main result achieved so far is the recognition of the content of ZSEE/RO and establishment of the necessary requirements for the access to the original data. In addition, many files have been converted to universal formats, totaling more than 500 (five hundred) converted files, and stored in the spatial database and available for consultants. Moreover, problems with regard to quality of the data have been detected, such as non-geocoded data, inconsistency of the data in different scales, and inconsistency of the available data in printed maps in rela-

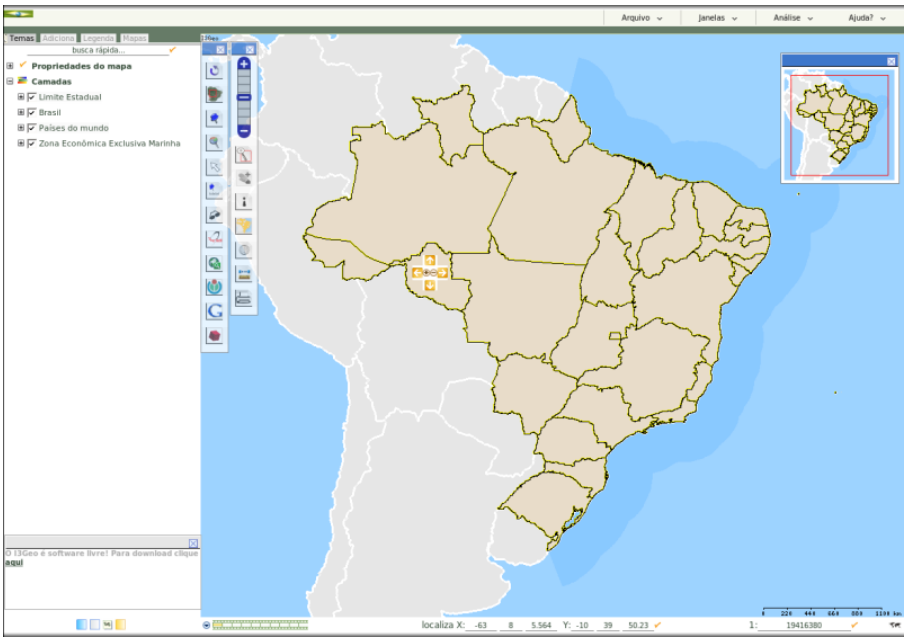


Figure 2: The I3Geo interface

tion to the stored ones in digital files.

Bibliography

[1] *Vulnerabilidade natural à erosão da Folha Presidente Mèdici* â Rondônia. 2005. 138p. Dissertação (Mestrado em Desenvolvimento regional e Meio Ambiente) Fundação Universidade Federal de Rondônia, Porto Velho.

[2] What is Geoserver. (online). 2007 http://docs.codehaus.org/display/GEOS/Home May 2007

[3] GeoNetwork opensource Community website. (online) 2007 http://geonetwork-opensource.org/ March 2007

[4] *Dos projetos de desenvolvimento, ao desenvolvimento dos projetos: o PLANAFLORO em Rondônia*. 2002. 285p. Tese (Doutorado em Ciências Humanas/Sociedade e Meio Ambiente) - Universidade Federal de Santa Catarina, Florianópolis.

[5] PostgreSQL Global Development Group, PostgreSQL 8.1.0 Documentation. (online) 2005 http://www.postgresql.org/docs/manuals/

[6] PostGIS Manual (online) 2005 http://postgis.refractions.net/documentation/ March 2007

[7] Projeto TerraView (online) 2007 http://www.dpi.inpe.br/terraview April 2007

[8] Wikibooks I3geo (online) http://pt.wikibooks.org/wiki/I3geo April 2007

Luis Fernando Bueno, Luiz Gilberto Dall'Igna
Sistema de Proteção da Amazônia - SIPAM
Avenida Lauro Sodrè, 6500
Aeroporto, Porto Velho, Rondônia
CEP 78903 - 711, Fone (69) 3217 6360
luis.bueno AT ipam.gov.br, luiz.dalligna AT sipam.gov.br

Marcelo Vitor Amaral Campos, Thiago de Lima Martarole, Diego Gomes Ferreira
Instituto Luterano de Ensino Superior de Porto Velho - ILES/ULBRA
Rua João Goulart, 666
Bairro Mato GrossoPorto Velho, Rondônia
CEP 78915 - 450
phone (69) 3216 7600
campos.mv AT gmail.com, shinoda.br AT gmail.com, diegogferreira AT msn.com

GeoSIPAM

Free and Open Source Software Applied to the Protection of the Brazilian Amazon

Luis Fernando Bueno, Weberson Gabriel, Pablo Filetti Moreira, Artur Henrique Villa Real F. Ramos, Fabio Augusto das Dores Silva, Marcelo Campos Brito

Keywords: *Spatial Data, Brazilian Amazon, SIPAM, Infrastructure, GeoSIPAM*

Abstract

This paper presents the infrastructure implemented and maintained by the System of Protection of Amazon - SIPAM for storage and distribution of spatial data called GeoSIPAM. The use of free and open source applications for the storage and distribution of the spatial data was chosen. The relational database PostgreSQL was chosen, with the PostGIS extension for spatial data storage and manipulation. Several applications (GeoServer, GeoNetwork and InterMap) have been customized for use by SIPAM. The implemented customizations involved procedure development which enhanced the tools used in daily tasks played by the collaborators. The application interface of the GeoNetwork and InterMap were altered aiming at making them accessible and adding features. Care was taken that the infrastructure was adjusted to spatial data storage, manipulation and distribution. The customizations allowed the applications to be merged into SIPAMś organizational environment. It was also verified that the use of OGC standards were complied with for interoperability on diverse systems.

Introduction

The database of the System of Protection of Amazon - SIPAM integrates information brought up to date on the Brazilian Legal Amazon. The use of these information in projects developed for the SIPAM and agency partners provides the generation of knowledge that assists the planning and coordination of global actions of government, aiming at the protection, social inclusion and sustainable development of the region.

This work presents the infrastructure implemented and maintained for SIPAM for storage and distribution of spatial data called GeoSIPAM. GeoSIPAM aims at providing integration and evaluation of data to aid the planning and the coordination of the actions of the developed public politics in the Brazilian Legal Amazon. Specifically, the objectives of GeoSIPAM are: to display through Internet or Intranet the metadata referring to the projects carried through for SIPAM; to make available Internet maps, geocoded images and related literal information; to visualize maps, images and relevant information for other institutions, through the use of standard compliant Open Geospatial Consortium - OGC (1) protocols

Free and Open Source Software in GeoSIPAM

The development and use of free software for geoinformatics has made available an increasing number of software tools. In the designing of GeoSIPAM it was opted to use free and open source software for the storage and distribution of the spatial data. The tools chosen include:

- PostgreSQL Object-relational database management system
- PostGIS Spatial database extension for PostgreSQL
- GeoServer OpenGIS Transactional Web Feature Server
- GeoNetwork Catalog application to manage spatially referenced resources through the web
- InterMap Map viewer, generally configured to operate of form integrated to the GeoNetwork

Infrastructure for Spatial Analysis

The chosen software architecture was organized in layers involving

- a Database Management System (DBMS) for data storage, management and manipulation
- maps and catalogues servers for data distribution and information
- as well as interfaces for data access based on the standards established for the OGC.

The PostgreSQL DBMS was chosen, with the PostGIS extension for storage and manipulation of

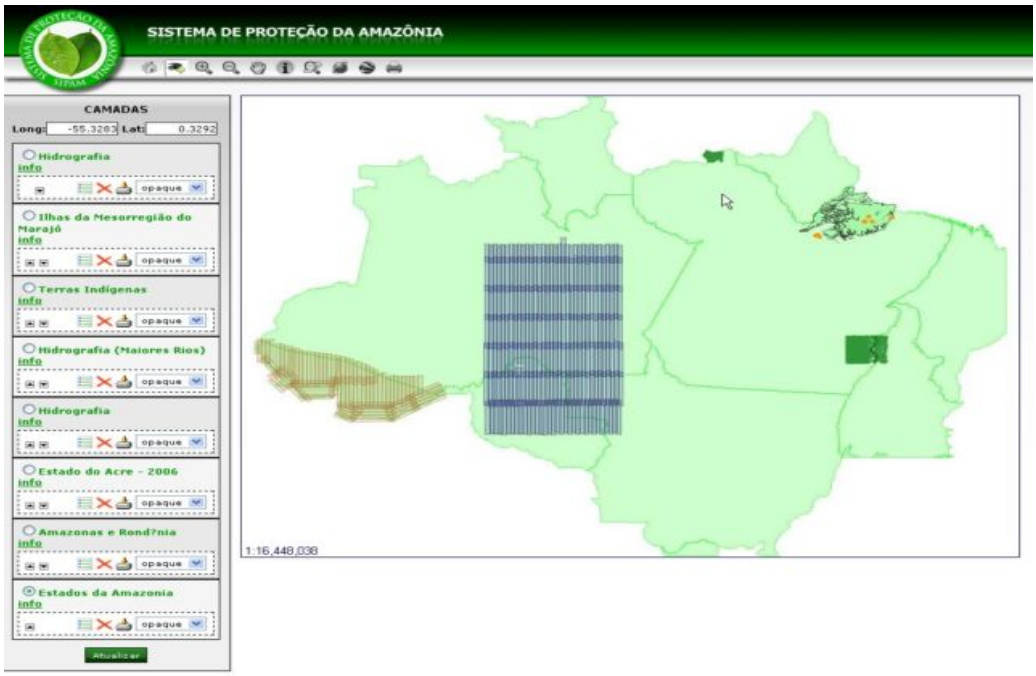


Figure 1: Interactive map visualization, usage of the customized InterMap software interface

spatial data. The GeoNetwork and InterMap applications interface had undergone alterations aiming to make them accessible and unique.

Figure 1 presents the visualization of an interactive map in the customized interface of InterMap software.

The GeoServer, GeoNetwork and InterMap, applications are tools developed in Java and implement the Web Map Services (WMS), Web Feature Services (WFS) and Catalogue Service Web (CSW) standards. These were customized for use at SIPAM. The implemented customizations involved development of procedures that enabled collaborators to carry out daily played tasks. Following are some customizations implemented by the GeoSIPAM team.

Originally InterMap Open Source used to attend the requirements of the standard WMS / OGC, but did not attend to the standard WFS / OGC, both offered by GeoServer. In InterMap Open Source a service was created to carry through WFS service solicitations to GeoServer, making it possible to download raster files. In this way, an option for the WFS GetFeature service call was developed, in the SHAPE - ZIP format and the service services.map.DownloadService was implemented. Also a corresponding button in the layer toolbar was created, as demonstrated in figure 2. This functionality was implemented by source code changes, involving JavaScript, XSL and Java programming.

```
public Element exec(Element params,
      ServiceContext context) throws Exception
{
// Get request parameters
int id = Integer.parseInt(
    params.getChildText(
        Constants.MAP_SERVICE_ID
    ));

// Get the MapMerger object from the user session
MapMerger mm = MapUtil.getMapMerger(context);

// Get the layer name
String nome = mm.getService(id).getName();
// Get the WMS server url
String serverUrl = mm.getService(id).getServerURL();

// Declare and set the WFS server url
int pos = serverUrl.indexOf("/wms")+1;
String WfsFeatureInfoUrl = serverUrl.substring(0, pos) +
Element el = WmsGetCapClient.getCapabilities(serverUrl);

// Return the necessary information for
// the service execution
return new Element("response")
.addContent(WmsGetCapClient.getCapabilities(serverUrl))
.addContent(new Element("serviceName").setText(nome))
.addContent(
  new Element(
```

```
    Constants.URL).setText(
      WfsFeatureInfoUrl));
}
```

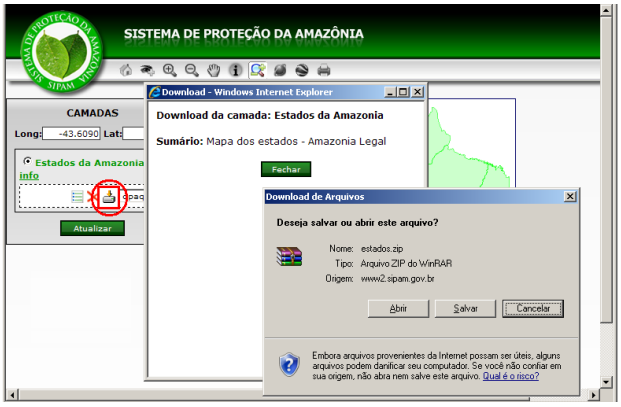


Figure 2: Customized InterMap interface for the download of layers

The option to offer the layer for download is configured through service WFS. The WFS GetFeature service allows download indiscriminately all the available layers in the server. The option was created to allow/deny the layer download in the cadastre screen of the FeatureType in the GeoServer. This information is also sent to WMS customer through the service getCapabilities. The Download button will be shown in the InterMap whenever the corresponding option is qualified in the GeoServer, as demonstrated in figure 3. The functionality was implemented through changes in the GeoServer source code, which is written in Java.

Another implemented functionality allows the visualization of the data together with images provided by Google Earth API. An inserted button in the InterMap toolbar sets in motion a floating window that shows the spatial data, synchronizing the GeoServerś raster data with the Googleś API images. The position and zoom are kept synchronized even after changes in the Google API window or InterMap maps' window. The InterMap layers are displayed in the Googleś API window in transparent shades, allowing simultaneous viewing of Googleś images. Figure 4 presents the Google API window implemented in InterMap. This functionality was implemented by the inclusion of JavaScript code in the file im_main.xsl. The code line below shows the reference to the Google API, in the file im_main.xsl:

```
<script
 src="http://maps.google.com/maps? \
    file=api&v=2&key=GOOGLE_API_KEY"
 type="text/javascript">
</script>
```

The term GOOGLE_API_KEY should be replaced by the key obtained on the Google API site[24]. To obtain the key you should access the site and provide the server URL where InterMap is installed. The key is sent to the email address of the applicant, who must have a GMail account. The functions "addWMS" and "montaMapa" were coded and added to the file im_main.xsl. The function "addWMS" is used to rebuild the list of layers to show the Google API, always when a new layer is added in InterMap. The function "montaMapa", is used to construct the Google map. Below is shown the source code of the functions "addWMS" and "montaMapa".

```
// Add WMS servers and layers types
function addWMS(servidor, camada){
 achou = false;
//Check if the server exists.
// If it does, add a new layer.
// If it does not register a new
// server and layer
for (contador = 0; wmsurl.length>contador; contador++){
if( wmsurl[contador] == servidor) {
// server found.. add layer
layersG[contador] = layersG[contador] + ',' + camada
achou = true;
}
}
if(!achou) {
// server not found, register in the vector
pos = wmsurl.length;
wmsurl[pos] = servidor;
layersG[pos] = camada;
}
listalayers = listalayers + camada
}

// Set up the google map on the screen
function montaMapa() {
var camadaWMS = new Array();
if( visivelG ) {
mapG = new GMap2(document.getElementById("mapaG"));

// A handler is called when a
// map movement break event is called
GEvent.addListener(mapG, "moveend", function() {
// If the movement was done at the Google map,
// the Intermap map changes too
if( actOrigem == '' ){
var center = mapG.getCenter();
// Obtain the center
main.map.zMapCenLat = center.lat();
// in LatLong
```

[24]Google API site: http://www.google.com/apis/maps/

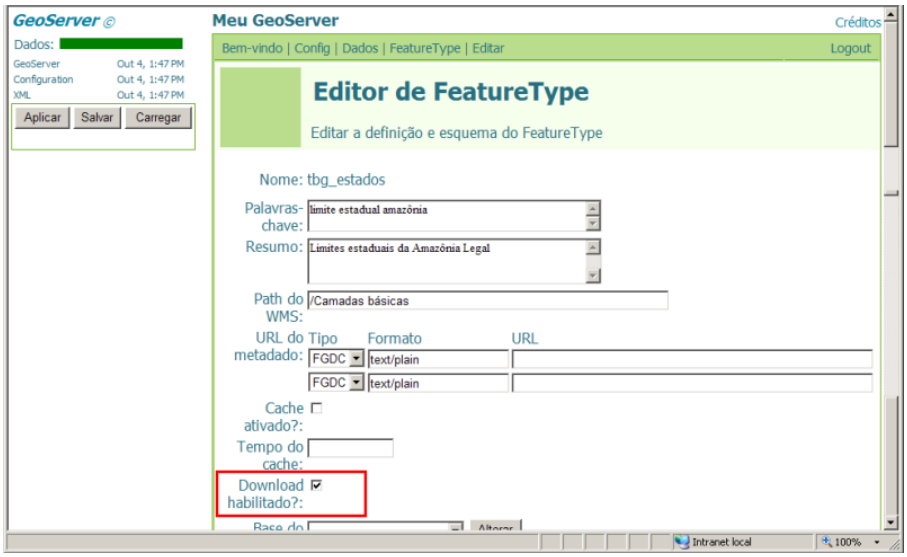


Figure 3: Customized GeoServer interface for enabling download

```
main.map.zMapCenLng = center.lng();
main.map.moveIntermap();
// moves the Intermap map
}
actOrigem = '';
if(navm){setTimeout("pngfix()",1000); }
});
// Handler is called
// when a drag action event is called
GEvent.addListener(mapG, "dragstart", function() {
actOrigem=''
});

// Handler is called
// when a zoom event is called
GEvent.addListener(mapG, "zoomend",  \
    function(zoomant, zoomatu) {
if( actOrigem == '') {
var bbox = mapG.getBounds();
// Obtain the center
main.map.zmapminlat = bbox.getSouthWest().lat();
// em latitude e longitude
main.map.zmapminlng = bbox.getSouthWest().lng();
// e move o mapa
main.map.zmapmaxlat = bbox.getNorthEast().lat();
main.map.zmapmaxlng = bbox.getNorthEast().lng();
main.map.zoomIntermap(zoomant, zoomatu);
// no intermap
}
if (actOrigem != 'X'){actOrigem = '';}
});

// Handler is called
// when a drag action event is called
```

```
var layer1=[G_SATELLITE_MAP.getTileLayers()[0], \
    G_HYBRID_MAP.getTileLayers()[1]];
var custommap1 = new GMapType(layer1, \
    G_SATELLITE_MAP.getProjection(), \
    "Google", G_SATELLITE_MAP);

var layer3=[G_SATELLITE_MAP.getTileLayers()[0], \
    G_HYBRID_MAP.getTileLayers()[1]];
for( contador=0; wmsurl.length > contador; \
contador++){
// create tile layers
camadaWMS[contador]= new GTileLayer( \
    new GCopyrightCollection(""),1,17);
if( wmsurl[contador] != '' )
{ if( layersG[contador] != '') {
camadaWMS[contador].myLayers=layersG[contador];
camadaWMS[contador].myBaseURL=wmsurl[contador];
camadaWMS[contador].getTileUrl=CustomGetTileUrl;
if(navm) {camadaWMS[contador].myFormat='image/png'}
} }
layer3[contador+2] = camadaWMS[contador];
}

var custommap3 = new GMapType(layer3, \
    G_SATELLITE_MAP.getProjection(), \
    "Intermap", G_SATELLITE_MAP);

mapG.getMapTypes().length = 0;
mapG.addMapType(custommap3);
mapG.addMapType(custommap1);
SWLatLng = new GLatLng(SWLat, SWLng)
NELatLng = new GLatLng(NELat, NELng)
bbox     = new GLatLngBounds(SWLatLng, NELatLng)
zoom     = mapG.getBoundsZoomLevel(bbox)
gmapt = mapG.getMapTypes()
```

```
zoomMax = \
    gmapt[1].getMaximumResolution(mapG.getCenter())
if( zoom > zoomMax) {zoom = zoomMax;}
actOrigem = 'X'
mapG.setCenter(bbox.getCenter(),zoom);
mapG.addControl(new GLargeMapControl());
mapG.addControl(new GMapTypeControl());
with(document.getElementById( \
    "google").style){top = "113px";left = "239px";}
actOrigem = '' } }
```

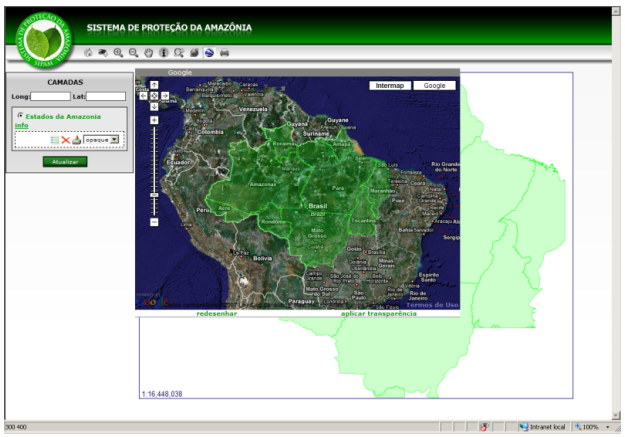


Figure 4: Customized InterMap software interface with Google API

Because of the great number of layers available in WMS servers, the list layers visualization, when shown as a flat list, becomes inadequate. The display of layers was modified to be loaded in tree form with options of "collapse" and "expand". The functionality was implemented by the inclusion of JavaScript code in the file im_get_services.xsl. It employed the dTree API, a free JavaScript tree menu. dTree JavaScript can be obtained online.[25]. In this case, the strategy adopted was entering the WMS server layers in an array. Thus, the layers tree is mounted by dTree from this array. Figure 5 demonstrates the layers visualization organized in tree form.

Results

An infrastructure was chosen and implemented that insured the capabilities for spatial data storage, manipulation and distribution. The database was populated with spatial data regarding the Legal Brazilian Amazonia, including topics such as hydrography, geology, use and occupation of land, mineral resources, administrative boundaries, roads, soil, etc. Spatial data from SIPAMś other partners, such as the National Agency of Water (ANA) and Brazilian Geological Service (CPRM), Brazilian Institute of Geography and Statistics, IBGE was also stored in GeoSIPAM. The volume of data stored grows every day. Local customizations to available FOSS GIS packages allowed implementing some features into SIPAM which were particular to organizational environment. It was also verified that the use of OGC standards ensures interoperability with diverse systems. The spatial data and products elaborated for the operational team of the SIPAM are being registered in a cadastre and stored in the GeoSIPAM system. This information is being distributed in the internal net and also for the public in general, through the SIPAMś web portal over the Internet. Among the main partners which constantly access the site are the Environment Secretaries of the State Governments of Amazonia, the Environmental Police, regional universities and others. Implemented statistics controls have logged more than 100,000 hits to GeoSIPAM.

[25] dTree JavaScript API: http://www.destroydrop.com/javascripts/tree/

Bibliography

[1] Bueno, L.F., P.G. Zuza e W. Gabriel. GeoSIPAM: Manual do Usuário. Presidência da República, CENSIPAM, CTO/Pv, 2007, 52 páginas.

[2] Câmara G.; C. Davis, e A.M.V. Monteiro. Introdução à Ciência da Geoinformação. (online). 2001. http://www.dpi.inpe.br/gilberto/livro/introd/. 14 April 2007

[3] Garnett, J. e C. Holmes. What is Geoserver. (online). 2007. http://docs.codehaus.org/display/GEOS/Home 10 May 2007.

[4] GeoNetwork opensource Community website. (online). 2007. http://geonetwork-opensource.org/ 19 March 2007.

[5] Landrö, Geir. dTree. (online). 2003. http://www.destroydrop.com/javascripts/tree/ 05 Setembro 2007

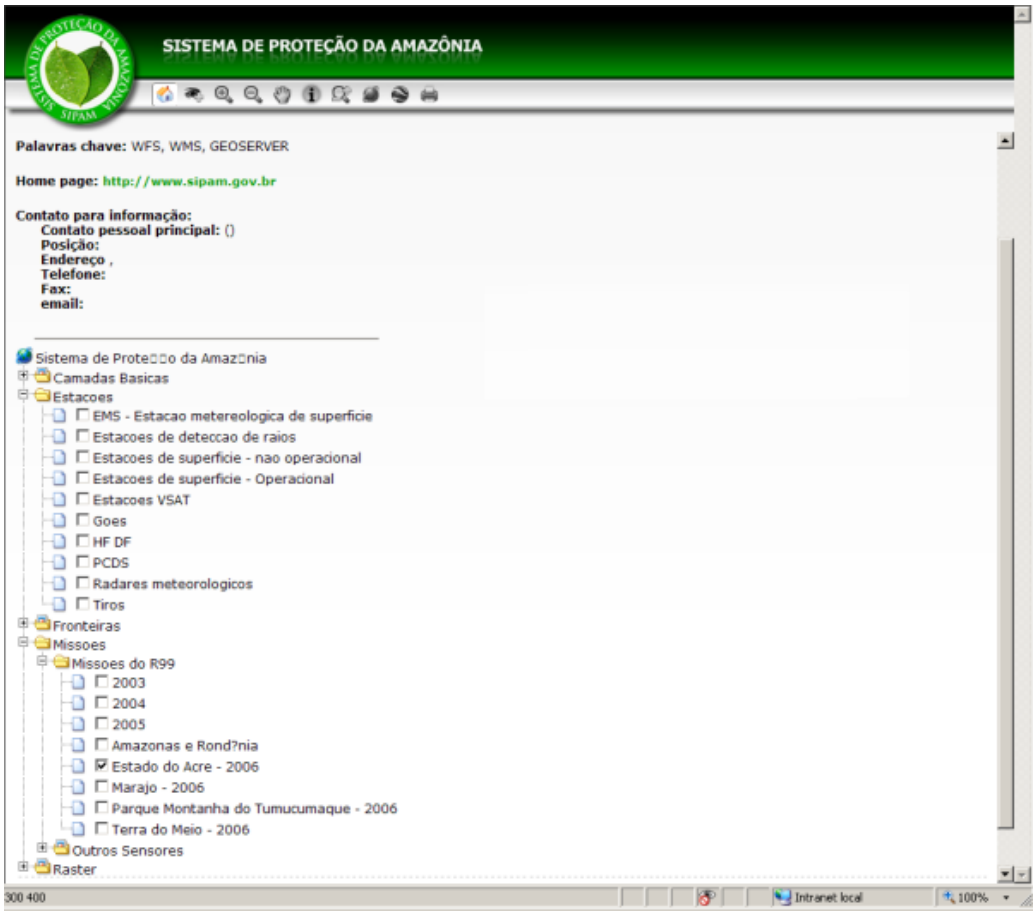


Figure 5: Customized InterMap interface with list layers vizualization

[6] PostgreSQL Global Development Group. PostgreSQL 8.1.0 Documentation. (online). 2005. http://www.postgresql.org/docs/manuals/ 05 May 2007

[7] Refractions Research. PostGIS Manual. (online). 2005. http://postgis.refractions.net/documentation/ 27 March 2007

[8] Wheeler, D.A. Why Open Source Software/Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers! (on line) http://www.dwheeler.com/oss_fs_why.html 03 June 2007

Luis Fernando Bueno
System of Protection of Amazon - SIPAM
SPO Área 05, Quadra 03, Bloco "K" Brasília - DF
CEP 70610 - 200
luis.bueno AT sipam.gov.br

Weberson Gabriel
weberson.gabriel AT sipam.gov.br

Pablo Filetti Moreira
pablo.moreira AT sipam.gov.br

Artur Henrique Villa Real F. Ramos
artur.ramos AT sipam.gov.br

Fabio Augusto das Dores Silva
fabio.silva AT sipam.gov.br

Marcelo Campos Brito
marcelo.brito AT sipam.gov.br

The Amazon Deforestation Monitoring System

A Large Environmental Database Developed on TerraLib and PostgreSQL

Freitas, U. M., Ribeiro, V. O., Queiroz, G. R., Petinatti, M. R. and Abreu, E. S.

Abstract

Brazil's National Institute for Space Research and the Foundation for Space Science, Technology and Applications developed a complete monitoring system, based on TerraLib open source technology[26] (Camara, G., et al., 2000), in order to map and calculate the annual deforestation rates in the Brazilian Amazon. TerraLib implements the archiving of geographic vector and raster data, on a variety of proprietary and non-proprietary DBMS, including PostgreSQL. TerraLib supports methods for image and vector data processing and analysis. A client application, named TerraAmazon, was developed using C++ and the free graphic user interface toolkit QT (version 3), which runs on LINUX or Windows machines. The data is managed by PostgreSQL version 8.2, running on a LINUX Server. The application manages all data work flow, gathering around 600 satellite images, pre-processing, segmenting and classifying these images, for further human interpretation and edition, on a concurrent multi-user environment. The database stores approximately 2 million complex polygons and 20 gigabytes of full resolution satellite images are added every year, using TerraLib pyramidal resolution schema. A Web site is provided for visualization and analysis of full resolution data, using the TerraLib PHP extension and TerraLib OGC WMS server.

Introduction

Brazil conducts a large environmental project to monitor deforestation in the Amazon biome using satellite data. Every year a deforestation map and the rate of yearly deforestation are produced and made public over the Internet by Brazil's National Institute for Space Research ("Instituto Nacional de Pesquisas Espaciais" – INPE). The Brazilian Amazon biome covers an area of 4.7 million square kilometers. Given this huge area, the task is very demanding. At every year a complete coverage of the region by satellite images, with 20 to 30 meters resolution, are acquired, automatically processed and analyzed by remote sensing specialists.

The final deforestation data product has cartographic precision suitable for a 1:250,000 scale. This project is named PRODES – short for Legal Amazon Deforestation Project - started at the end of the 1980s, and has evolved from an analog interpretation process to a fully digital procedure. The current methodology was implemented in 2005 and its technical features are presented in this paper.

Before the new system became operational, deforestation maps were produced using SPRING, a free desktop image processing and geographic information system developed by INPE.[27] In order to produce the complete deforestation map, 229 independent databases, each one covering the area of one LANDSAT 5 satellite image were required. This methodology created a complex environment for management since each database was transferred from one workstation to another to be submitted to a specific process, involving dozens of specialists.

In addition to the complexity of the previous methodology, new requirements forced the Brazilian government in 2005 to improve the former methodology. The first new requirement was the need to introduce multi-satellite source, in order to guarantee data availability, even under a satellite operational interruption. Images from the 20 meter resolution CBERS (China Brazil Earth Resources Satellite) CCD sensor, 30 meter LANDSAT 5 images and 32 meter DMC (Disaster Monitoring Constellation) satellites images are now used. Figure 1 shows the satellite images used for 2005 deforestation mapping. The second requirement was the need for a fast data delivery, in order to create conditions to implement government policies to be applied earlier, before the next period of deforestation.

The use of CBERS images as the primary data source increases the number of images to 570 and the use of the previous methodology with independent databases would have created a yet more complex environment for data management. The solu-

[26] TerraLib Library: http://terralib.org/
[27] INPE SPRING project: http://www.dpi.inpe.br/spring

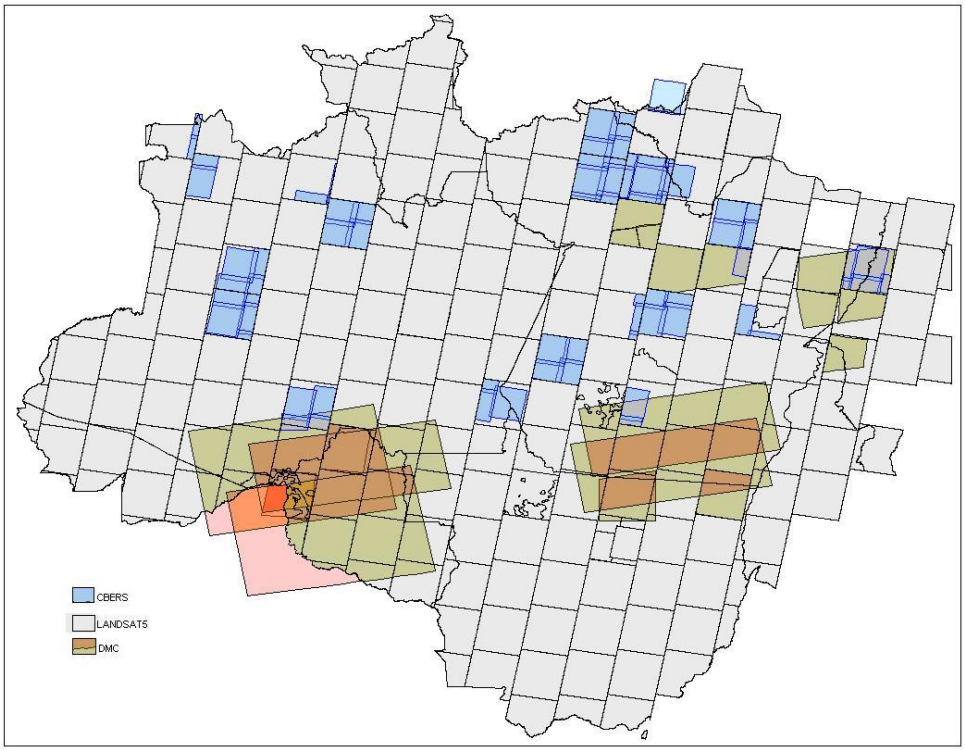


Figure 1: The 221 CBERS, 223 LANDSAT, and 18 DMC satellite images used in 2005

tion was to create a unique corporate database which is suitable for management of all data operations, in a distributed and concurrent environment.

The technology selected to achieve the project goals was TerraLib. TerraLib is an Open Source Library developed by INPE and distributed under GNU LGPL license.[28] TerraLib implements the storage of geographic vector and raster data, on a variety of proprietary and non-proprietary Database Management System (DBMS), including PostgreSQL. TerraLib implements methods for image and vector data processing and analysis. FUNCATE under contract with INPE developed the complete suite of computer programs to process all data and deliver the deforestation maps and annual rate, on a full open source environment. This suite of programs was named TerraAmazon.

Methodology

Deforestation and subsequent burnings occur in Amazon during a short period: the dry season, from July to September. After this season, it is virtually impossible to deforest, due to the high rates of precipitation. Based on this fact, the annual deforestation rate is calculated for the period between August 1st of the previous year and July 31st of the current year. The later date coincides with the end of the dry season for most part of the Amazon. In order to obtain the deforestation rate, images acquired during the dry season period are analyzed. The annual deforestation rate is estimated by interpolation, considering that deforestation occurs linearly during the dry season. In addition, deforested areas are estimated under regions covered by clouds, considering that the ratio of deforestation is the same in areas with and without clouds coverage. Detailed information of this method can be found at INPE's site.[29]

TerraAmazon manages all operations required by the deforestation project, in an interactive, distributed and concurrent environment using a corporate database. In order to take full advantage of TerraAmazon, the whole Amazon region is divided into cells and each cell is manipulated by only one remote sensing specialist at a given time.

Cells are created by partitioning the project extents using a geographic grid with 0.25 degrees dis-

[28] TerraLib Library: http://www.terralib.org
[29] Detailed information available online: http://www.inpe.br/prodes

tance between grid lines. Each remote sensing specialist can lock one or more cells to process using a long transactions schema. The expert manipulates one of these cells using image processing and vector edition tools available in TerraAmazon. The following steps are applied to each of these cells.

1. Import a TIFF image
2. Register image with reference image and save used control points
3. Audit image using reference image, and
4. If image is not approved then repeat from step 2
5. Create shade and ground images
6. If image has cloud coverage above a given threshold then
   (a) Classify image to extract regions with clouds
   (b) Convert regions to cloud polygons
7. Segment shade and ground images
8. Combine ancillary vectors (previous years deforestation, non forest, and hydrography polygons), segmentation polygons, and cloud polygons (if any)
9. Interpret and edit combined vectors to create new deforestation and cloud polygons
10. Audit resulting polygons. If not approved then return to 8; and
11. Disseminate results.

The image processing tools available in TerraAmazon are: TIFF format image file import, georeferencing based on control points, color composition and enhancement, mixing model analysis, segmentation, and classification.

For vector edition, TerraAmazon include: raster to vector and vector to raster conversion, vector elements edition that considers snap and topology, and set operations (union, difference, intersection, and overlay) operations on geographical features.

Other TerraAmazon tools include visualization and database check{}-in and check{}-out procedures.

TerraAmazon Implementation Details

TerraAmazon is a database client application, developed on top of TerraLib geographic components library, using Standard C++ programming language and graphical interface implemented using the free graphic user interface library QT.[30] TerraAmazon can be executed on LINUX or MS-Windows environments. All data is managed by the PostgreSQL DBMS[31], running on a LINUX server.

Each of the 0.25 degrees cell is blocked by the remote sensing specialist in a long term transaction schema, bounded by check-in and check-out operations. The cell field is used to clip all available geographic representations in order to reduce the amount of geographic elements, guaranteeing manageability of graphic features. For fast visualization these graphic features are cached in memory and indexed by a linear R-Tree (Guttman, 1984). Figure 2 shows a region of Amazon with cell edges highlighted in green.

TerraAmazon topological restriction operations are used during the edition of new deforestation areas and clouds. Before a new deforestation or cloud polygon is stored in the database, TerraAmazon subtracts from the polygon previous deforestation polygons. Figure 3 shows these steps for a new cloud polygon.

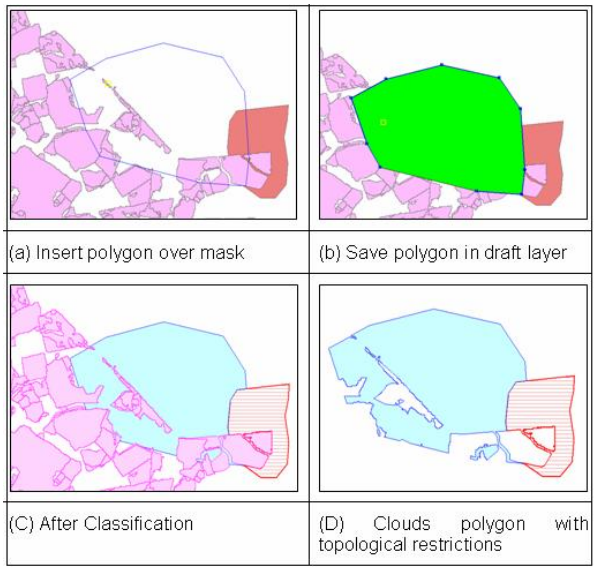


Figure 3: TerraAmazon topological restriction applied to a new cloud polygon. The polygon stored in the database is the one presented in (D).

The complete set of TerraAmazon tools is composed by:

1. Import of TIFF image;
2. Georeference using polynomial model calculated from control points;

[30] QT Library: http://trolltech.com
[31] PostgreSQL DBMS: http//:www.postgresql.org

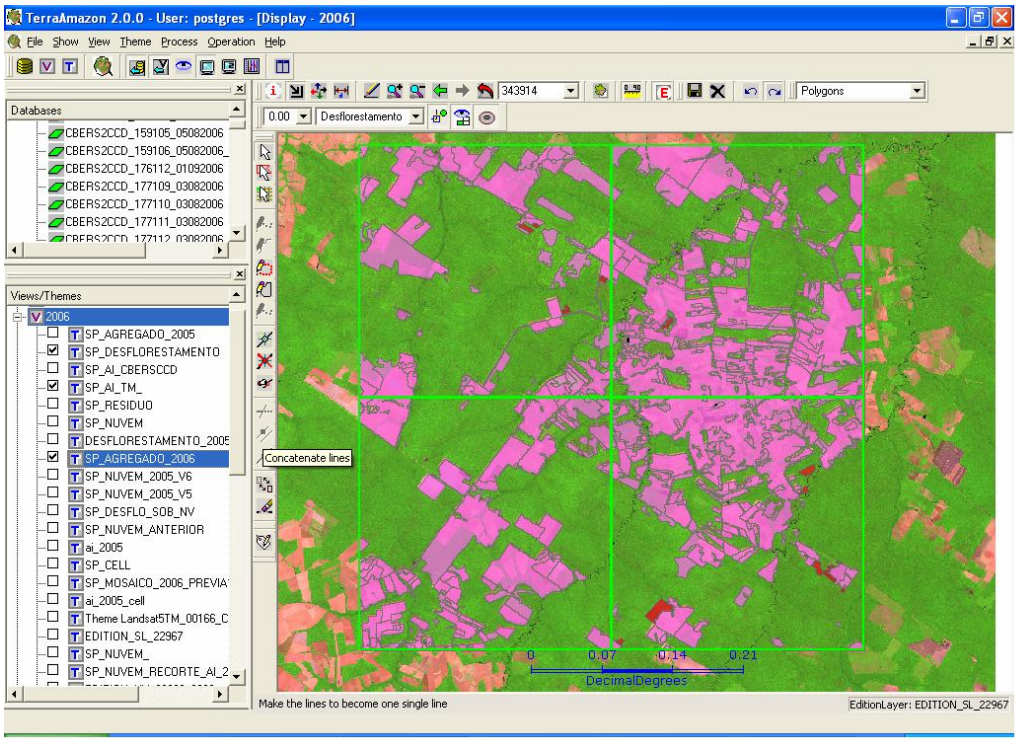


Figure 2: Cells (highlighted in green) selected for edition in a small part of Amazon.

3. Image enhancement and color composition;
4. Mixing model analysis; (Shimabukuro,1987); (Shimabukuro;Smith,1991)
5. Image segmentation using region growing algorithm; (Bins et al., 1996)
6. Labeling of regions;
7. Clustering classification;
8. K-means classification;
9. Raster to vector and vector to raster conversion;
10. Graphic interface for vector edition with snap and topological control;
11. Union, difference, intersection, and overlay set operations on graphical features;
12. Check-in and check-out procedures using cells index.

An Internet distribution site is provided, based on a PHP application running on a LINUX Web Server, powered by Apache. The Web application was created on top of TerraLib library, using the TerraPHP extension.

The Internet site feature includes:

1. Seamless visualization of full resolution data;
2. Image visualization, using pyramidal resolution;
3. Export of full resolution features, defined by user;
4. Web Map Server - WMS - access to data;
5. User queries, including deforestation by municipalities and inside protected areas;
6. Deforestation ranking by municipality;
7. Deforestation indices by cell grid;

Figure 4 shows the Internet dissemination site[32].

Deforestation Project Figures

The following figures demonstrate the huge task made possible by TerraAmazon:

- To create the deforestation map for the period 2004-2005, 221 CBERS images, 223 LANDSAT images and 18 DMC images was used, for 2005-2006, 70 CBERS images and 211 LANDSAT images
- During the interpretation phase, in 2006, the system was accessed by up to 20 concurrent

[32]Internet dissemination site: http://www3.funcate.org.br/prodes2

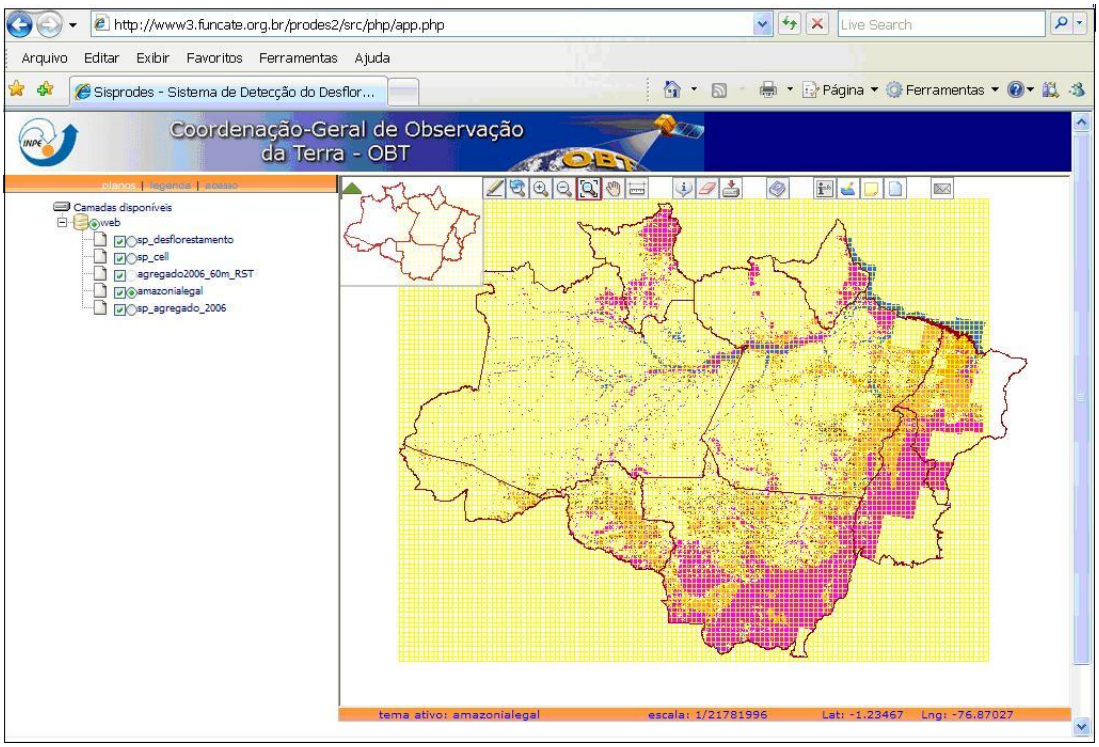


Figure 4: TerraAmazon Internet deforestation data dissemination site.

users. These users added 213,693 new deforested polygons and 595,575 new cloud polygons to the database

- The final database stores 2,380,880 polygons, classified in different categories. These polygons are complex - the largest one has 69,925 vertices, the average number of vertices is 59 and the average number of holes per polygon is 7
- The current volume of data stored in database is 237GB and includes full resolution images using a multi-resolution pyramidal RLZ compressed schema

Conclusion

TerraAmazon fulfilled the requirements imposed by the Brazilian Government and has been used since 2005. TerraLib technology is an option to create a complex GIS system, using only open source software and made feasible the implementation of TerraAmazon. In addition, TerraAmazon has been proving to be robust, easy to manage and reliable in a high demand production multi-user environment.

Acknowledgments

This work was supported by INPE - National Institute for Space Research. The authors wish to thank the development and user groups at FUNCATE for their efforts. Their collaboration is really greatly appreciated. We are grateful to Dr. Laercio Massaru Namikawa for his support and suggestions.

References

Gutmman, A. R-trees a dynamic index structure for spatial searching. *ACM SIGMOD International Conference on Management of Data*, 47-57, 1984.

[Shimabukuro,Y.E.(1987)] Shade images derived from linear mixing models of multispectral measurements of forested areas. Ph. D. Dissertation, Department of Forest and Wood Sciences, Colorado State University, Fort Collins, Colorado. 274p.

[Shimabukuro,Y.E.;Smith,J.A.(1991)] The least-squares mixing models to generate fraction images derived from remote sensing multispectral data. IEEE Transactions on Geoscience and Remote Sensing, v. 29, n. 1, p. 16- 20

[Bins et al.(1996)] L. Bins, L. M. Fonseca, G. J. Erthal, F. Mitsuo Ii (1996) Satellite Imagery Segmentation: a Region Growing Approach;Anais do VIII Simpsio Brasileiro de Sensoriamento Remoto: Salvador

[Camara, G., et al. (2000)] TerraLib: Technology in Support of GIS Innovation; II Brazilian Symposium on GeoInformatics, GeoInfo2000. São Paulo

Vanildes Ribeiro, Ubirajara Freitas, Gilberto Queiroz, Mario Petinatti, Eric Abreu
FUNCATE - Fundação de Ciência, Aplicações e Tecnologia Espaciais
http://www.funcate.org.br
vanildes AT funcate.org.brg

The Open Source Geospatial Foundation, or OSGeo, is a not-for-profit organization whose mission is to support and promote the collaborative development of open geospatial technologies and data. The foundation provides financial, organizational and legal support to the broader open source geospatial community. It also serves as an independent legal entity to which community members can contribute code, funding and other resources, secure in the knowledge that their contributions will be maintained for public benefit. OSGeo also serves as an outreach and advocacy organization for the open source geospatial community, and provides a common forum and shared infrastructure for improving cross-project collaboration.

Published by OSGeo, the OSGeo Journal is focused on presenting discussion papers, case studies and introductions and concepts relating to open source and geospatial software topics.

Acknowledgements
Various reviewers & the GRASS News Project

The *OSGeo Journal* is a publication of the *OSGeo Foundation*. The base of this journal, the LaTeX 2_ε style source has been kindly provided by the GRASS and R News editorial board.

Journal online: http://www.osgeo.org/journal
OSGeo Homepage: http://www.osgeo.org
Postal mail: OSGeo
PO Box 4844, Williams Lake,
British Columbia, Canada, V2G 2V8
Telephone: +1-250-277-1621

ISSN 1994-1897