Technology Products of the PHAiRS REASoN Project – Year 1

Karl Kent Benedict[ab], William Hudspeth[a]
[a]Earth Data Analysis Center,University of New Mexico
MSC01 1110
1 University of New Mexico
Albuquerque, NM 87131
[b]Corresponding Author: kbene@edac.unm.edu

***Abstract*— New technology development for the Public Health Applications in Remote Sensing (PHAiRS) NASA REASoN project consist of two components: 1) development of modularized visualization tools that integrate model outputs, statistical methods and additional geospatial reference information; and 2) the development of automated data acquisition, processing, and integration technologies that streamline the ingestion of NASA data and other model outputs into the dust generation and propagation model being regionalized for the DSS as part of the Research component of our REASoN project. A combination of Open Source technologies are being used to meet the requirements of both components. The visualization component integrates MapServer (an Internet mapping application), R (a mathematical programming and statistical analysis package), and GRASS (a GIS application) through custom Python scripts to present to the user a combination of mapping and analysis tools via a web interface. All four software environments have been successfully integrated into a client interface and the interface is currently undergoing testing and evaluation. Data acquisition and processing automation has been accomplished through a combination of Python and Bash shell scripts. Shell scripts process MODIS land products, acquired as multiple HDF files, into mosaicked ArcASCII Grids for ingestion into the dust model, GRASS rasters for analysis, and GeoTIFF's for data download. Python scripts automatically acquire current NCEP ETA forecast data via OPeNDAP and reprocess these data into GeoTIFF and ArcASCII grids, both for use as dust model inputs and as data available for analysis and visualization within the user interface. In total these technology developments within the project facilitate the timely acquisition of data and provide the tools and interfaces needed to improve public health decision-making within the Rapid Syndrome Validation Project (RSVP) DSS.**

***Index Terms*— Public Health, Remote Sensing. Decision Support.**

I. Introduction

THE year one technological accomplishments of the Public Health Applications in Remote Sensing (PHAiRS) project, funded under NASA's REASoN program[1], have moved the project towards the twin goals of: 1) providing enhanced web-based visualization and analytic tools to public health decision makers relating to environmental factors influencing human health; and, 2) infusing NASA data and model outputs into the Dust Regional Atmospheric Model (DREAM) [1]–[3] in order to provide enhanced dust forecast capabilities for delivery to public health officials.

These activities relate to an overall goal of developing an application framework that enhances the current capabilities of the Rapid Syndrome Validation Project (RSVP) Public Health decision support system (DSS)[2], a system developed and maintained by researchers at Sandia National Laboratory in cooperation with Los Alamos National Laboratories, the University of New Mexico Department of Emergency Medicine, and the NM Department of Health Office of Epidemiology, and the University of New Mexico's Earth Data Analysis Center. Ultimately these enhancements of RSVP will be accomplished through the development of three related technologies. First, map image services that may be directly integrated into RSVP's user interface. Second, through the development of a free-standing, web-based, interactive mapping environment that allows users to explore and analyze environmental and aggregated public health data. Third, the development of technologies that streamline the ingestion of new environmental data into the system, both as model inputs for DREAM and for presentation to end users. This paper describes the progress made in the latter two technology areas.

II. Visualization and Analytic Tools

Web client development at the Earth Data Analysis Center has converged in recent years on a unique combination of integrated tools and applications to produce powerful and dynamic mapping applications. At the core of these efforts is the development of a modularized block of programmatic code written in the Python scripting language. Bearing the moniker Mapmodule, this code block imports and accesses the functionality of a variety of available Python modules to provide web-based mapping clients with spatial database functionality, raster processing and analysis capabilities, and the ability to leverage statistical analysis and plotting routines on spatial data.

Minnesota Mapserver, developed at the Environmental Resources Spatial Analysis Center (ERSAC), Department of Forest Resources, University of Minnesota (UMN) as part of a

[1]This work is supported by NASA, Cooperative Agreement No. NNS04AA19A

[2]All relevant web addresses are provided in Appendix I

cooperative effort with the Minnesota Department of Natural Resources, is an open source development environment for constructing spatially enabled Internet-web applications. The MapServer C API supports the display of a wide variety of vector and raster formats using the OGR Simple Features and Geospatial Data Abstraction (GDAL) libraries. It also provides support for feature selection by item/value, point, and area, TrueType fonts, map element automation (scalebar, reference map, and legend), scale dependent feature drawing and application execution, feature labeling, on-the-fly projections, and thematic map building. Map composition is configured in a static map file where layer content, display characteristics, projection parameters, and other map attributes are defined in a standardized, hierarchical format. The MapServer API can then be used to dynamically modify various map file attributes, allowing a user to effect such changes as layer visibility, map scale, and feature selection in the displayed map. All such changes are effected when the Common Gateway Interface (CGI) executes user requests submitted to the Mapserver API and returns a single image file of the map that reflects user-defined modifications.

The MapServer system includes Mapscript that allows popular scripting languages such as PHP, Perl, Python, and Java to access the MapServer C API. Development at the Earth Data Analysis Center has experimented with the PHP scripting language in the past, but is now almost exclusively focused on the Python version of Mapscript. Developed and currently supported by Sean Gillies at Zcologia, Python Mapscript provides access to the MapServer API largely through a series of classes, allowing the programmer to manipulate virtually all aspects of a map file, draw entire maps, layers, or individual shapes, perform spatial queries using points, areas, or other features, read/write shapefiles, and perform attribute queries. Using the Python programming language allows the developer to script Dynamic Hypertext Markup Language (DHTML) as a wrapper for integrating the Python Mapscript module. User interaction with map functions is thus enabled via standard form elements in the web mapping client.

As noted above, the Minnesota Mapserver API supports the display of a wide variety of vector and raster formats using the OGR and GDAL geospatial libraries. Development efforts at the Earth Data Analysis Center currently rely on two additional external applications to store and manage vector and raster geospatial datasets, respectively. Vector data is stored within a PostgreSQL/PostGIS framework. PostgreSQL is a highly scalable, SQL compliant, open source object-relational database management system. PostGIS adds support for geographic objects to the PostgreSQL database, allowing the latter to be used as a backend spatial database for geographic information systems (GIS), much like ESRI's Spatial Database Engine (SDE). The spatial enabling of PostgreSQL by PostGIS permits EDAC developers to take advantage of complex spatial and topological relationships between geographic objects in the database. It permits such functions as querying, overlay, and proximity types of operations. Programmatic access to the PostgreSQL/PostGIS database is enabled through the psycopg.py PostgreSQL database adapter for the Python programming language, developed at initd.org.

Raster datasets are managed, stored and processed in GRASS GIS. GRASS is a Geographic Information System used for geospatial data management and analysis, image processing, graphics/map production, spatial modeling, and visualization. Raster datasets from a wide variety of sources can be imported into the standard GRASS internal format of a matrix of values. These data can be re-projected, mosaicked or tiled, clipped, classified, and processed with any number of system- or user-defined pixel processing algorithms. Once imported into GRASS, rasters can be directly referenced and visualized by the Minnesota Mapserver API. Because GRASS geospatial processing operations are freestanding functions, they can be programmatically accessed from Python scripts using the OS family of Python operating system functions (os.py).

Statistical analysis of GRASS rasters, as well as the generation of graphical plots of the results of such analyses, are accomplished with the R project statistical package. Developed at Bell Laboratories, R is a free software environment for statistical computing and graphics. It provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, etc.) and graphical techniques, and is highly extensible. In fact, a multitude of packages have been developed to support analysis in many scientific domains. The GRASS: Interface between GRASS 5.0 geographical information system and R package, developed by Roger Bivand at the Norwegian School of Economics and Business Administration, provides functions to directly access GRASS data objects, and to import them into R data structures for subsequent analysis. Programmatically, R functions are accessed from Python code with Rpy, a module that provides a Python interface to R. As such, statistical analysis of rasters involves passing user-defined spatial extents (whether a point, transect, or area) to GRASS to extract the relevant pixel information into a separate layer. At this point, R language functions can be used to generate statistical information about a collection of points, and to plot this information for user interpretation.

To illustrate the programmatic flow of this integrated web-mapping system, we can deconstruct several standardized user operations into the steps involved in their implementation. The DHTML used in building the actual web client is written out with Python script. When a user first opens the client, they are presented with a map that reflects the current state of attribute settings in the static map file (Figure 1). In addition to the map, there are a variety of user-accessible control elements that can be used to manipulate the map and layer attributes in the map file. As discussed above, any given mapping client will begin with a static Mapserver map file that configures the basic properties of the map. When a user interacts with one of the form control elements in the client, this information is passed via the standard CGI form variables where it is evaluated programmatically. Requested changes to the map (e.g., changes in extent or the visibility of different layers) are processed with functions found in Python Mapscript to manipulate attribute values in a cached version of the mapfile. At this point, the Mapserver API can return a new image of the map that reflects the changes requested by the user (Figure 2).

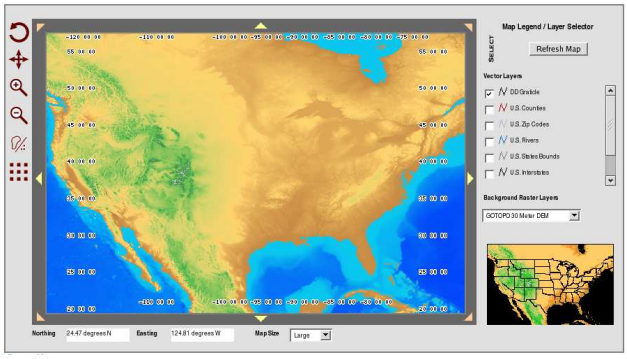


Fig. 1. Initial Client View

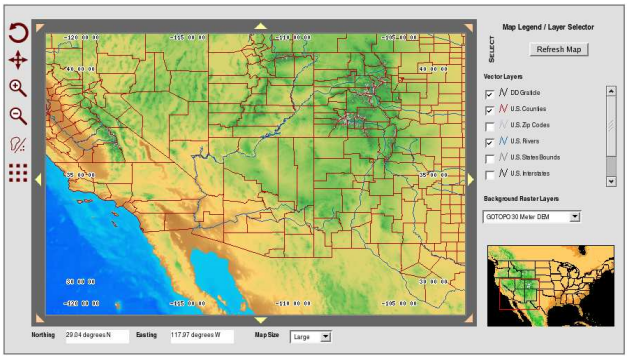
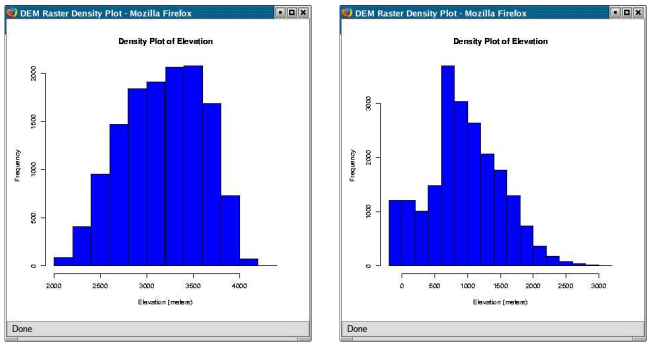


Fig. 2. Client View After Zoom and Activation of Additional Layers

More advanced functionality, such as querying vector layers, or analyzing subsets of a raster layer, require that the user specify the layer of interest, and then interactively select the objects or data subsets they are interested in. This frequently involves submitting geographic extents that define the location of the target objects to the Mapmodule-based client. These extents are then passed as specific kinds of functional requests to the appropriate component, whether PostGIS databases or GRASS functions, to process and return appropriate results (e.g. Figure 3). Programmatically, these requests and responses are translated across the rPy and psycopg connectors between Python and the target component.



(a) Southern Colorado (b) Death Valley, CA

Fig. 3. Two Density Plots of Elevation Raster Analysis Requests from Client Interface.

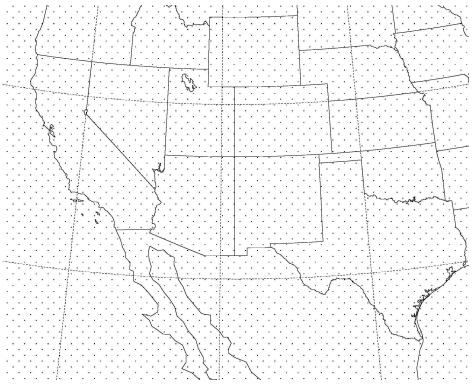


Fig. 4. DREAM Model Domain for PHAiRS

III. Data Ingest and Processing

A critical requirement for the successful implementation of an enhanced Public Health DSS is the availability of environmental and reference data that provides both the context for and characteristics of environmental factors that may influence human health. These environmental data may be derived from a variety of sources, and the emphasis in the early data ingest and processing capability developments has been on the processing of remote sensing data products from the National Aeronautics and Space Administration (NASA) and meteorological forecast outputs from the National Oceanic and Atmospheric Administration (NOAA). Specifically, the data products addressed in the work reported here include NASA's Moderate Resolution Imaging Spectroradiometer (MODIS) land products and NCEP/Eta meteorological forecast data.

MODIS land products and the NCEP/Eta forecast data were selected for the initial development efforts because both of these data sources have potential for or are currently used as initialization and/or boundary condition parameters in the DREAM model. In particular, the MODIS MOD12Q1 Land Cover product [4] has potential as a replacement for the Olson World Ecosystem 10-minute resolution vegetation data [5] currently used as a source of vegetation data, while DREAM currently uses multi-level NCEP/Eta forecast data for specific Humidity (Q), Temperature (T), U-Velocity (U), and V-Velocity (V) both for model initialization and as boundary condition parameters at 6-hour intervals. The capabilities developed thus far facilitate the generation of mosaicked MODIS land products that encompass the project model domain (Figure 4), and acquire NCEP/Eta data from the NCDC NOMADS4 GrADS Server (http://nomads.ncdc.noaa.gov:9090/index.html) via the OPeNDAP protocol automated through a custom Python script. The following sections outline the procedures and methods employed in achieving these results.

A. MODIS Land Product Processing

Processing of MODIS Land Cover data (MOD12Q1) [4] for use by the DREAM model requires four steps:

1) Mosaicking of multiple MODIS tiles into a single file for further processing

2) Reprojection from the provided sinusoidal projection into the decimal degree geographic coordinate system used by DREAM
3) Subsetting the mosaicked mosaic to the area of interest (Figure 4)
4) Conversion of the subsetted data into a format compatible with DREAM

These steps have been accomplished through a combination of MODIS and raster processing tools, with the entire process automated through a BASH shell script. The processing tools used include two programs provided as part of the MODIS Reprojection Tool (developed by with support from the Land Process Distributed Active Archive Center): mrtmosaic and resample. Further processing is performed using GRASS GIS.

The processing script accepts a single parameter at execution time, the name of the configuration file to use. The configuration file contains the names of the MODIS HDF files to process. All other processing instructions are embedded in the BASH shell script. Upon execution, the shell script verifies that all of the specified HDF files exist and after verification, runs the mrtmosaic program to mosaic the listed HDF files into a single HDF file. The MODIS Reprojection Tool resample program is then run on the resulting HDF file to both reproject and convert the file into a GeoTIFF. To achieve coverage over the entire model domain, 14 MODIS granules needed to be processed. The processed granules include:

- MOD12Q1.A2001001.h11v04.004.2004149203108.hdf
- MOD12Q1.A2001001.h11v05.004.2004149203110.hdf
- MOD12Q1.A2001001.h11v06.004.2004149203131.hdf
- MOD12Q1.A2001001.h10v06.004.2004149203022.hdf
- MOD12Q1.A2001001.h10v05.004.2004149203002.hdf
- MOD12Q1.A2001001.h10v04.004.2004149203002.hdf
- MOD12Q1.A2001001.h09v06.004.2004149202907.hdf
- MOD12Q1.A2001001.h09v05.004.2004149202901.hdf
- MOD12Q1.A2001001.h09v04.004.2004149202901.hdf
- MOD12Q1.A2001001.h08v06.004.2004149202758.hdf
- MOD12Q1.A2001001.h08v05.004.2004149202758.hdf
- MOD12Q1.A2001001.h08v04.004.2004149202802.hdf
- MOD12Q1.A2001001.h07v05.004.2004149202727.hdf
- MOD12Q1.A2001001.h07v06.004.2004149202725.hdf

If DREAM could ingest a GeoTIFF file as a raster data source, processing could stop at step 2), but DREAM is currently able to ingest only XYZ and ArcASCII gridded data, neither of which is supported by the resample program as a output format. Further processing to obtain gridded data in these formats is accomplished using the raster processing capabilities of GRASS GIS.

Processing of the GeoTIFF produced by the resample program in GRASS begins with setting the GRASS region to the decimal degree region defined for the project. This has the effect of defining the analysis area, and the limits of the area for which data will be processed. Importing the mosaicked GeoTIFF into GRASS will automatically subset the resulting GRASS raster to the limits of the defined region, reducing the processing requirements for the subsequent steps, and meeting the goals of step 3) above. After conversion to GRASS' internal raster format, the subsetted MODIS data may be exported into any of the numerous external raster formats supported by GRASS. In this case, both XYZ ASCII text and ArcASCII grids are exported for use by DREAM. Both formats were generated as a demonstration of the substantial reduction in file size that can be obtained by using the internal matrix representation used by the ArcASCII grid format in comparison with the XYZ format previously supported by DREAM for the import of gridded data. The export and automated compression of the resulting data files comprise the completion of the fourth and final step listed above.

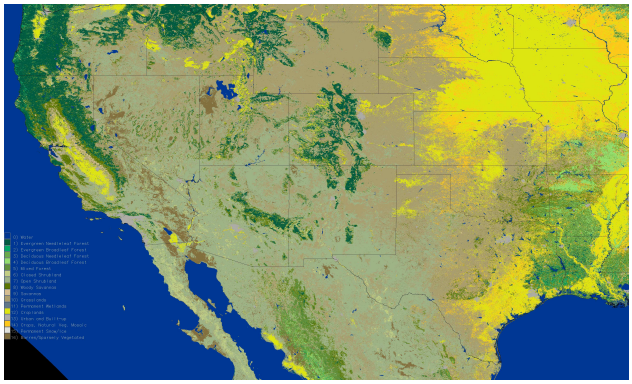


Fig. 5. Preview Image of Mosaicked MODIS Land Cover Granules

An additional benefit of importing the mosaicked MODIS data into GRASS is that additional representations of the data may be generated that are useful in visualization and validation. In particular, a preview image of the resulting grid, with source and classification information, is generated as an additional product of the script. This Portable Network Graphics (.png) file (Figure 5) can be easily viewed, printed, or included in documents or reports that describe the products of the project.

Overall, the development of a generalized automated processing method for MODIS Land Products has set the stage for efficient acquisition and ingestion of any Land Products that are needed by DREAM or for the visualization environment described above. This automated process runs in less than two minutes on the 14 MODIS granules listed above, providing a very efficient processing stream for data products entering the PHAiRS project data repository.

B. NCEP/Eta OPeNDAP Data Source Access and Processing

Though early in the development process, the automated acquisition and processing of NCEP/Eta forecast data via the OPeNDAP protocol represents a capability with great potential, both in reference to the specific data source (NCEP/Eta) under initial consideration, and in general. This potential exists because of the large number of datasets available as OPeNDAP services [6], including a wide variety of NCEP meteorological products. As with the development of the generalized processing script for MODIS Land Products described above, the development of generalized scripted access to OPeNDAP data sources vastly expands the number and variety of data and model outputs available for the PHAiRS project.

Progress thus far in the development of the automated acquisition and processing of NCEP/Eta data via OPeNDAP

has resulted in a Python script that successfully queries the specified OPeNDAP server for available datasets, retrieves OPeNDAP data into Python data objects, and outputs those data objects to the terminal for verification. This functionality has been achieved through use of the OPeNDAP/DODS Python Module developed by Roberto De Almeida. This module supports both client and server functionality, though only the client implementation is used in this implementation.

Further development of this capability is progressing through the implementation of file export functions for the script that will support the generation of ArcASCII and GeoTIFF grids for import into DREAM and presentation within the visualization and analysis environment. It is anticipated that this functionality will be achieved by the end of September, 2005.

IV. Conclusion

Overall, the progress made in the first year of the New Technology component of the PHAiRS project reflects both a rapid development cycle and closely linked integration of these new technologies into the products of the PHAiRS project. During this year of work, the visualization and mapping capabilities of the client interface have evolved from a basic browsable map interface based upon PHP mapscript to a Python-based modular multifunctional client that has already demonstrated statistical processing of raster datasets and has clearly defined workpaths that relate vector data (in PostgreSQL/PostGIS), raster data storage and processing (in GRASS), and statistical analytic capabilities (using R). At the same time, the project has made substantial progress in the development of automated ingestion and processing tools that will streamline the integration of new data and model outputs into the routine DREAM model runs and the visualization and analysis environment.

Appendix I
Resources

RSVP: http://www.ca.sandia.gov/chembio/implementation_proj/rsvp/
Minnesota Mapserver: http://mapserver.gis.umn.edu/
Python Mapscript: http://zoologia.com/mapserver/?r=1
GRASS GIS: http://grass.baylor.edu/
PostgreSQL: http://www.postgresql.org/
PostGIS: http://postgis.refractions.net/
psycopg: http://initd.org/projects/psycopg1
Python: http://www.python.org/
GDAL: http://www.remotesensing.org/gdal/
OGR: http://www.remotesensing.org:16080/gdal/ogr/
R Statistical Computing: http://www.r-project.org/
Rpy: http://rpy.sourceforge.net/download.html
GRASS 5.0-R Interface: http://sal.uiuc.edu/csiss/Rgeo/index.html
MODIS Reprojection Tool: http://edcdaac.usgs.gov/landdaac/tools/modis/index.asp
OPeNDAP/DODS Python Module: http://opendap.oceanografia.org/

Acknowledgments

The authors thank all of our partners on the PHAiRS team, including S. Nickovic, D. Yin, B. Chandy, B. Barbaris, A. Budge, T. Budge, S. Baros, C. Bales, C. Catrall, S. Morain, G. Sanchez, W. Sprigg, and K. Thome. The development of the MapModule has also been significantly assisted through support provided by Sean Gillies, the current maintainer of Python Mapscript.

References

[1] S. Nickovic, G. Kallos, A. Papadopoulos, and O. Kakaliagou, "A model for prediction of desert dust cycle in the atmosphere," *J. Geophys. Res.*, vol. 106, pp. 18 113–18 130, 2001.
[2] D. Westphal, O. Toon, and N. Carlson, "A case study of mobilization and transport of saharan dust," *J. Atmos. Sci.*, vol. 45, pp. 2145–2175, 1988.
[3] ——, "A two-dimensional investigation of the dynamics and microphysics of saharan dust storms," *J. Geophys. Res.*, vol. 92, pp. 3027–3049, 1987.
[4] National Aeronautics and Space Administration, "Modis/terra land cover type yearly l3 global 1km sin grid," http://edcdaac.usgs.gov/modis/mod12q1v4.asp, 2005, web page accessed on 5/15/2005.
[5] J. Olson, J. Watts, and L. Allison, "Carbon in live vegetation of major world ecosystems," 1982, released on CD as part of the Global Ecosystems Database, Ecosystem and Global Change Program, National Geophysical Data Center, Boulder, Colorado 80303, USA; 1994.
[6] OPeNDAP, Inc., "Opendap-accessible datasets," http://www.opendap.org/data/datasets.cgi?xmlfilename=datasets.xml&exfun%ction=none, 2005, web page accessed on 5/15/2005.

Karl Benedict Dr. Benedict's technical background includes 18 years hands-on computer solution implementation experience including: relational database design and administration; information needs assessment; network design, implementation, and administration; system specification, configuration, and administration; end-user support provision; multiple system and application integration; and Internet application development employing a combination of commercial and open-source server applications based upon Perl, PHP, Python, ColdFusion, Zope and Plone. Dr. Benedict has worked with applications running under AIX, Linux, Microsoft Windows, and the Macintosh OS, and currently is a Senior Research Scientist at and manages the IT program for the Earth Data Analysis Center at the University of New Mexico.

William Hudspeth Dr. Hudspeth has nine years experience in Geographic Information Technology and has spent the last three years participating in developing various applications that employ web-based database connectivity, display and analysis of geospatial datasets, interactive web-based mapping, and data delivery. This work has involved both proprietary and open source software and includes Cold Fusion, PHP, Python, Grass GIS, ArcGIS, Minnesota Mapserver, and R.