

Open-Source Geographic Information Systems

David A. Garbin and James L. Fisher

Enterprises can benefit from an open-source GIS architecture, but implementation success depends critically on understanding the nature of GIS data and the functions of the various open-source components.

As newspaper and e-zine headlines attest, climate change, severe weather, natural disasters, and terrorist actions are prominent subjects in current events—particularly where they occur and how they affect the local population. Of utmost concern is the ability to predict and prepare for such events, and in this regard, sophisticated computer models are proving invaluable. Integral to such analytic modeling are geographic information systems (GISs), which serve both as a means for analyzing and forecasting effects and as a medium for conveying information in an easily understood format that can reach the most users.

Indeed, it is not an exaggeration to say that GISs are becoming a framework for understanding a dynamic world. Users can overlay tropical storm prediction patterns from the National Weather

Service on roads and infrastructure operations centers to know what areas are most likely to be affected. They can plot proposed road routes over maps of designated wetlands and endangered animal safe zones to better understand the impact of potential pollution. And with good hydrology models, they can predict the impact of storm surges on waterfront commercial buildings to better determine how many roads and businesses must close to ensure public safety.

The scope of GIS implementation is also extensive, being anything from a desktop implementation to an enterprise operation with significant investment in data storage and processing assets.

It is not surprising then that a large commercial GIS industry has emerged to provide software tools, training, and source data to customers. Such commercial offerings are nearly commonplace. However, open-source applications have an equal potential for large-scale implementation across a broad range of platforms, and they can be far more cost effective and offer more flexibility. The success of such an implementation depends critically on selecting the right components and understanding their functions.

In April 2007, Noblis began implementing a GIS from open-source components for use in evaluating flood inundation and network resilience. Our implementation and work with other organizations in establishing GIS capability underlines the importance of informed preparation in assembling an open-source GIS environment.

Inside Track

- Open-source applications have an equal potential with commercial packages for large-scale implementation across a broad range of platforms—and they can be far more cost effective and flexible.
- An enterprise-level system should have at the very least a geodatabase, sophisticated server-based GIS tools, and Web-based publishing and visualization support.
- By its nature, geospatial data can be sensitive or proprietary to the project it supports, so any system must incorporate a strong security mechanism for the central repository.
- Desktop clients generally do not have extensive geoprocessing tools, but they excel at classifying and presenting data in an easily understood (dashboard) format.

Required components

As Figure 1 shows, the four main component classes for an enterprise implementation are

- enterprise-class geographic databases,
- scalable server-based geographic processing capabilities,
- Web-based publishing tools that produce results anyone can display and share, and
- desktop client tools for visualization and interactive editing and analysis—either browser-based or more functional GIS clients.

Geodatabases are familiar relational database management systems with additional geographically oriented data types and spatial operators that enable queries like “select all municipalities within a given radius of a specific latitude and longitude.” For each entity of interest, such as a state or country, geodatabases hold data values (population count and rainfall amount, for example), geographic coordinates, and the coordinate system in which the geographic coordinates are expressed. The entity

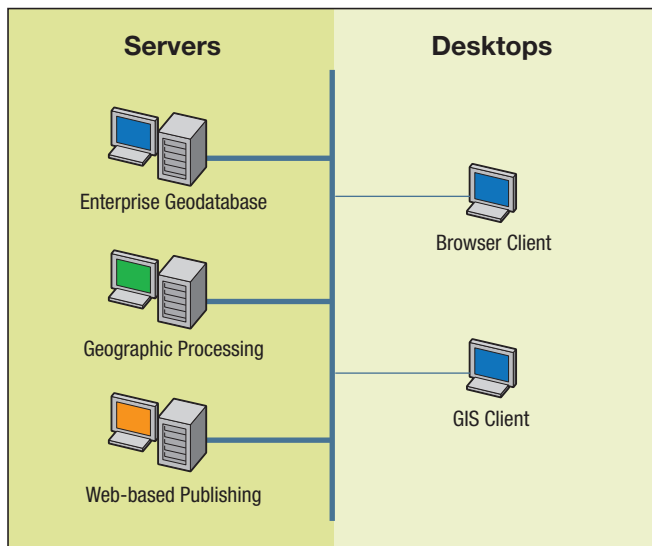


Figure 1. Enterprise GIS components. Server-based components do the heavy lifting, while user desktops provide flexible tools in the form of thin and thick clients. All these components are readily available from open sources and run on a variety of server and desktop operating systems.

might be a point for a city, a line segment for a road, or an area for a geopolitical boundary. The geodatabase holds only data that the GIS processes directly. Data such as satellite images, which the system typically uses only for displaying background images to help the user’s visual orientation, is stored in flat files.

By its nature, geospatial data can be sensitive or proprietary to the project it supports, so any system must incorporate a strong security mechanism for the central repository. User authentication, specific user roles, and table-level restrictions should be part of any enterprise-level database.

Because processing geospatial data typically requires many spatial computations, a *central server* with lots of memory and

fast processors is desirable. Personal desktop computers running standalone GIS software are adequate for smaller applications, but enterprise-scale problems require the addition of server-based processing, memory, and storage capabilities.

Often those who want to understand the results of a geospatial analysis do not need a full suite of tools to perform the underlying geoprocessing, nor do they need any background maps and data sets for geographic areas outside their area of interest. *Web-based publishing tools* bridge the gap between background and results by producing sharable output. Such output can be either a single image of a complete map or features that the user can display on top of third-party maps from such applications as Google Maps or Google Earth.

Last but by no means least are *visualization tools*, which can range from full-function GIS desktop applications with a rich set of geospatial processing capabilities to view-only browsers. Geospatial data is in layers, with each layer representing a particular feature that can be overlaid on other features to present a comprehensive view of a geographic area. Each feature, in turn, can have one or many associated attribute values. A county feature might have a vector polygon representing its boundaries, and its attributes might be population, area, and annual budget. A given GIS layer can display one of these attributes, classify the values into groups if desired, and display them with various options (different symbols for points, varying line thickness, and different color polygons, for example). Desktop clients provide this functionality in a user-friendly manner with a legend for all layers that is easy to view.

Being able to view data in a projection different from that of the underlying feature layer is important because source data can come in many projections. Instead of converting each dataset into a common projection, desktop clients import each layer in its native format and convert the layers to a common projection for display. Even if all source data is in a common projection, say, a Mercator projection for the United States, users might prefer a different projection, such as one that avoids area distortions.

Data types and sources

Before an enterprise can effectively use the open-source tools for manipulating geospatial data, it must understand both the types of data it will be acquiring and how best to provide mechanisms for ensuring that the data formats are compatible.

Data types

All GIS data is either raster or vector, differing in storage requirements and allowable operations.

Raster data represents continuous geographic phenomena, such as elevation and temperature, using a cell or pixel matrix to represent a specific numeric value. Each pixel is uniform and

represents a specific point in space. The smaller the cells, the greater the accuracy, but also the greater the file size. If the raster values represent light levels, they basically present a picture of the geographic feature of interest. The most popular variant is visual satellite imagery; other forms include elevation data, which is useful in showing river and ocean bed topologies and in other remote sensing applications. An important feature of rasters is that users can mathematically combine data from multiple rasters that cover the same area, deriving new information that they can then display as a raster. Because raster data does not compress well, requiring vast amounts of hard disk storage space, they are not typically stored in databases.

Vector data, in contrast, uses points, line segments, and closed polygons to represent a wide range of real-world features efficiently, including points of interest, roads, and geopolitical regions. Vector data is more common in geospatial processing, in part because each data element represents a discrete object, so users can assign attribute values to that object and store them in a database along with the object's geometry.

File formats

GISs exchange geospatial data through files. The most popular data format for exchanging vector data is the Environmental Systems Research Institute's (ESRI's) shapefile—a set of files describing the location, shape, and other attribute information about geographic features. For exchanging point data, standard ASCII files containing latitude and longitude values are sufficient.

Raster data in any number of standard formats is suitable for images. However, geospatial information about the raster, such as the coordinates of the starting pixel, the projection used, the pixel's horizontal and vertical resolution, and the standard pixel values must augment the raster data. Some formats require embedding such information in the file itself; other formats require a supplementary file. Typical file formats for exchanging raster data include GeoTiff, JPEG2000, and NetCDF.

A third set of file formats is based on the Extensible Markup Language, the most popular being the Keyhole Markup Language (KML), which serves as input to Google Earth. Users specify vectors directly in KML and specify rasters by naming the image file and including geospatial information in the KML entries.

Because users normally have little control over the input data's format, any GIS should be "multilingual," and most GISs—both commercial and open source—can read and write a range of popular file formats. Output formats are mostly a matter of local preference and knowing the likely user community for the data.

Coordinate systems and projections

Map projections represent information from a curved surface as a two-dimensional projection. Projections fall into several classes, the most common being cylindrical, conic, and azimuthal. Each class can be conformal, equal-area, or neither.

Table 1 shows the most common projections used for general-purpose maps. The particular projection depends on the project's purpose and the size, shape, and location of the area of interest. As a rule, conformal projections are more appropriate for visualizing country-wide maps, since these projections preserve angular relationships and are better at preserving arc length. The familiar depiction of the United States in maps is the Lambert conformal conic projection, for example. Equal-area projections are more appropriate for statistical studies and work in which the amount of material is important.

Table 1. Applying the most common map projection classes.

Projection class	Projection type	Description
Cylindrical	Neither	Areas that span more than 6,000 miles east to west. An example is the Mercator World Map.
Conic	Conformal	Mid-latitude regions where direction and distance are important. An example is the Lambert conformal conic projection of the United States.
Conic	Equal-area	Mid-latitude regions where depiction of area is important. An example is the Albers equal-area projection of North America.
Azimuthal	Neither	Polar regions. An example is the azimuthal equidistant polar projection.

Precise mathematical relations define all projections, and such relations must also govern any method of projecting coordinates from a geographic reference frame (latitude-longitude) into a projected Cartesian reference frame. GISs perform these transformations automatically using standard libraries, but the software must be aware of the projection for any imported geographic object, regardless of the format used.

Combining sources

One of the most difficult aspects of assembling a GIS environment is ensuring that all the formats, coordinate systems, and projections are compatible, particularly local files that must merge with national coverage files. Applications such as municipal planning analysis often commingle local and national sources. National coverage alone can be too coarse for such applications. The U.S. Geological Survey provides geospatial data layers of roads, railroads, airports, states, counties, forests, waterways, and dams, and elevation data is available in raster files. The U.S. Census Bureau produces road, county, and census statistical data in shapefiles as part of the Topologically Integrated Geographic Encoding and Referencing (Tiger) map project.

In contrast, many major cities have detailed GIS data of their transportation systems, building placement, and waterways, captured as a combination of detailed images and geographically

marked civil engineering drawings in vector format. Many public and private industries, such as utilities, maintain restricted geospatial data sets typically in local coordinate systems, such as the Universal Transverse Mercator Zone or Federal Information Processing Standards State Plane projections. Enterprises must take care when combining these files with national coverage files.

Open-source community

Like any other open-source community, the open-source geospatial community consists of hundreds of companies and individual contributors working toward a common goal. Support for this community comes mainly from two organizations. The Open Source Geospatial Foundation (OSGeo) is a not-for-profit organization that aims to support and promote the collaborative development of open geospatial technologies and data. The Open Geospatial Consortium (OGC) is an international industry consortium of more than 300 companies, government agencies, and universities attempting to develop publicly available interface specifications.

Technologies and data

In addition to developing technologies and data, OSGeo provides financial, organizational, and legal support. OSGeo projects develop major open-source GIS software and the standards-based libraries that enable interoperability among packages. Figure 2 shows how OSGeo organizes its projects. Of particular importance is its provision of the Geospatial Data Abstraction Library/OpenGIS Simple Features Reference Implementation (GDAL/OGR), a cross-platform translator library for raster and vector geospatial data formats. As an open-source library, it presents a single abstract data model to the calling application

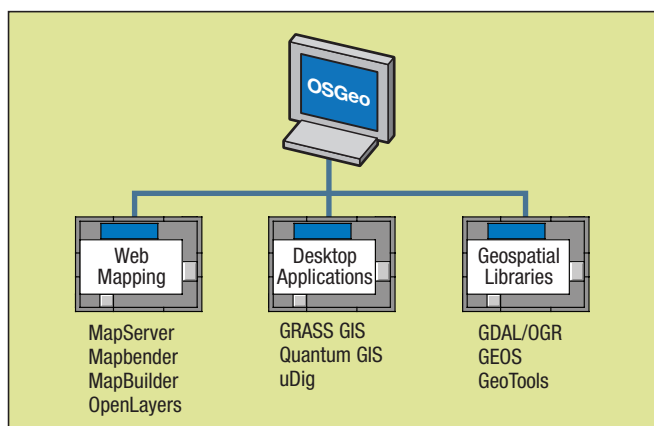


Figure 2. Open Source Geospatial Foundation (OSGeo) projects. Within this structure, OSGeo develops major open-source GIS software and the standards-based libraries that allow packages to interoperate. OSGeo projects cover the spectrum of GIS components needed for an enterprise architecture. The OSGeo website (www.osgeo.org) contains project descriptions in each category.

for all supported formats. It also comes with a variety of useful command-line utilities for data translation and processing. GDAL supports 50+ raster formats, and OGR supports 20+ vector formats. Other libraries, such as Geometry Engine–Open Source (GEOS), provide additional types of processing tools.

Standards

The OGC's primary contribution is the OpenGIS Specifications, which support the interoperable solutions that equip the Web, as well as wireless and location-based servers with the ability to process geospatial data. Important components of the OpenGIS specifications are the Web Feature Service (WFS), Web Mapping Service (WMS), Geography Markup Language, Sensor Model Language, and Simple Features. These standards promote interoperability both among open-source tools and open-source and commercial GIS software. For example, with WMS, simple browsers can view fully rendered maps, while WFS provides geospatial features that desktop GIS clients can use as layers.

A sample environment

Figure 3 shows the Noblis GIS environment, which uses open-source tools that map to the components in Figure 1. Specifically, the implementation has five main parts—all of which are open source:

- PostgreSQL, a relational database system, extended with PostGIS to support geographic objects;
- Geographic Resources Analysis Support System (GRASS), software for data management, image processing, graphics production, spatial modeling, and data visualization;
- MapServer, a development environment for building spatially enabled Internet applications;

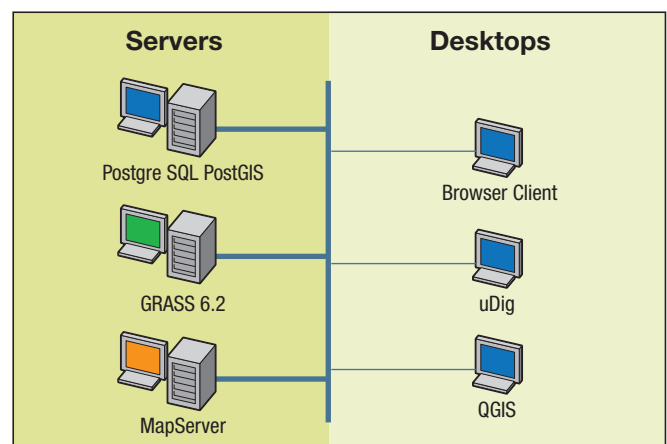


Figure 3. The Noblis open-source GIS environment.

- User-friendly Desktop Internet GIS (uDig), a dedicated desktop GIS client that provides the flexible visualization of geospatial data suitable for presenting analytical results; and
- Quantum GIS (QGIS), a dedicated desktop GIS that features integration with GRASS objects and tools.

Our implementation uses Debian Linux multiprocessor servers for the geodatabase and GIS software. User machines running Windows XP access the servers using open-source X-Windows clients. Kerberos and SSL technologies assist in maintaining privacy and security.

Geodatabase system

PostgreSQL is a proven architecture with more than 15 years of active development. It runs on all major operating systems, including Linux, Mac OS X, Unix, and Windows, and can accommodate large amounts of data and many concurrent users. The system runs stored procedures in more than a dozen program-

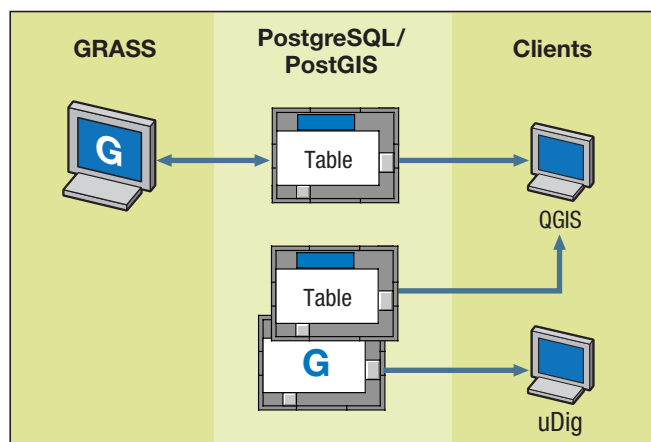


Figure 4. The PostgreSQL and PostGIS components of Noblis' GIS architecture. Geographic information (G) is held in the PostGIS data records or stored in files using the native GRASS format. QGIS can visualize PostGIS tables as well as the combined GRASS/PostgreSQL objects.

ming languages, including Java, Perl, Python, Ruby, Tcl, C/C++, and its own PL/pgSQL.

PostGIS enhances PostgreSQL with support for geographic objects. In effect, PostGIS lets the PostgreSQL server be used as a backend spatial database for GISs, much like ESRI's Spatial Database Engine or Oracle's Spatial extension. As Figure 4 shows, user clients can visualize spatial objects stored in PostGIS directly, or GRASS can use PostgreSQL as a database to store attribute data, while GRASS stores the topology. With this arrangement, GRASS has access to the full range of SQL commands and functions in PostgreSQL, which it can use to manipulate data and control what is displayed.

Data management and processing

Developed in 1982 by the U.S. Army Construction Engineering Research Laboratories (a branch of the U.S. Army Corp of Engineers) for land management and environmental planning, GRASS has evolved into a powerful utility with broad application in many scientific research areas. GRASS has a stellar reputation for flexibly processing raster data, and with version 6, it added equally significant capabilities for vector data processing. Unlike most vector GIS programs that use the Simple Features vector model, where shapes are just shapes, GRASS stores a full topology that models nodes, links, and their relationships. Consequently, the system can process the flow of commodities through networks—be they water in a riverbed, cars on a road, or packets in a communications network. Using OGR library modules, GRASS can import and export its topological model to formats compatible with external programs.

Besides using the power of PostgreSQL to handle attribute data, GRASS enables data visualization on its own, although with limited features. Features normally associated with user-friendly graphical interfaces are lacking, such as on-the-fly data reprojection. However, in an enterprise architecture, programs

Component Sources

These sources, described in more detail in the main text, are readily available to any enterprise seeking to build a GIS:

Open-source projects and standards organizations

- The Open Source Geospatial Foundation (OSGeo); <http://www.osgeo.org/>
- Open Geospatial Consortium (OGC); <http://www.opengeospatial.org/>

Data sources and formats

- Environmental Systems Research Institute's (ESRI's) shapefile technical description; <http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf>
- Keyhole Markup Language (KML) documentation; <http://code.google.com/apis/kml/documentation/>
- U.S. Census Bureau's Topologically Integrated Geographic Encoding and Referencing (Tiger) map project; <http://www.census.gov/geo/www/tiger/>
- U.S. Geological Survey geospatial data layers; <http://nationalatlas.gov/maplayers.html>

Data management and processing

- Geographic Resources Analysis Support System (GRASS); <http://www.osgeo.org/grass>

Geodatabases

- PostgreSQL; <http://www.postgresql.org/>
- PostGIS; <http://www.postgis.org/>

Browsing and rendering

- MapServer development environment; <http://mapserver.gis.umn.edu/>

Desktop clients

- Quantum GIS; <http://www.osgeo.org/qgis> and <http://www.qgis.org/>
- User-friendly Desktop Internet GIS (uDig); <http://udig.refractor.net/>
- GE Graph; <http://www.sgrillo.net/googleearth/gegraph.htm>

such as QGIS and MapServer can interface directly with GRASS objects to provide the missing capabilities.

Browsing and rendering

MapServer excels at rendering spatial data (maps, images, and vector data) for the Web. Beyond browsing GIS data, it lets users create geographic image maps—maps that direct interested parties to content. The program was developed by the University of Minnesota's ForNet project in cooperation with the National Aeronautics and Space Administration and the Minnesota Department of Natural Resources.

As input, MapServer supports all GDAL/OGR formats, including direct support for PostGIS database objects. Most important, it provides Web services using OGC standards, such as WFS and WMS, and supports thousands of on-the-fly reprojections. Through WFS, MapServer can be a service-oriented architecture for clients that need full GIS functions served over the Web. Any Web browser can access finished maps as images, which means that a larger population can share the results.

Desktop clients

Desktop clients generally do not have extensive geoprocessing tools, but they excel at classifying and presenting data in an easily understood (dashboard) format. Both QGIS and uDig can thematically present data layers and offer on-the-fly reprojection of geographic data.

QGIS runs on Linux, Unix, Mac OS X, and Windows and supports vector, raster, and database formats. One of its most salient features is its ability to visualize GRASS objects directly as layers. QGIS can run on the desktop directly or on the server along with GRASS accessible through X-Windows.

uDig is an application framework built with Eclipse Rich Client technology that aims to provide a complete Java solution for desktop GIS data access, editing, and viewing. It runs natively on Windows, Mac OS X, and Linux systems and directly supports all standard formats, including full integration with PostGIS and support for Web geospatial services. Enterprises can use uDig as a development platform for building standalone, customized user applications.

Google Earth

Google Earth provides access to high-resolution imagery over the whole globe, and its built-in feature layers provide a wealth of content as well as a significant capability for users to add their own layers using KML files. Users can convert the output of GRASS or any other geoprocessing software, including both vector and raster formats, to KML files and deliver these to Google Earth clients on the Web. In the raster image in Figure 5, Noblis analysts predicted the absolute water level using a finite-element computer model; GRASS then computed the flood depth over land using accurate elevation data as input, and provided geocoded image files using KML. Analysts overlaid these onto Google Earth images.

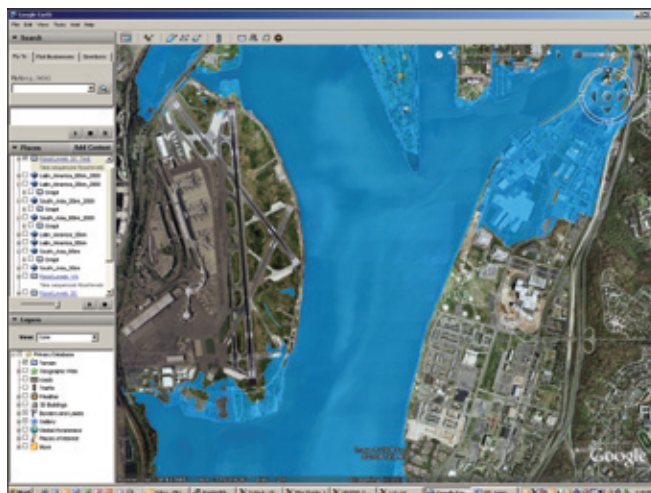


Figure 5. Flood inundation in the Washington, D.C., area from Hurricane Isabel. GRASS created the original raster image of flooded areas using output from a computer model and accurate ground elevation data. Analysts were then free to import the results into Google Earth as a semi-transparent ground overlay.

GE Graph adds graphing capabilities to Google Earth. Given Excel spreadsheet data, the program provides complete KML files.

The tools we have described hint at the rich diversity inherent in both GIS applications and in the open-source tools being developed. Using these sources as a departure point, any enterprise should be able to tailor a GIS environment that will continue to serve a variety of processing needs for the foreseeable future.

Perhaps most important—and what gives open source its power—is in the supporting community, where user feedback drives innovation. As part of the open-source community, an enterprise is no longer tied to a particular vendor and can enjoy the perspectives and expertise from a vast storehouse. Such feedback is good insurance against the gaps in functionality and platform changes that often plague commercial offerings. ■



David A. Garbin is a senior fellow at Noblis, where his interests include telecommunications technology, networking, network design and optimization, economic analysis, and voice and data communications. He received an MSEE from the Massachusetts Institute of Technology. Contact him at david.garbin@noblis.org.



James L. Fisher is a senior principal engineer at Noblis, where his interests include the design of open and encrypted network communications. He received a PhD in electrical and computer engineering from Carnegie Mellon University. Contact him at jlf@noblis.org.