Enterprises can benefit from an open source geographic information system architecture, but understanding the nature of GIS data and the functions of each open source component is crucial to implementation success.

Climate change, severe weather, and natural disasters 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; in this regard, sophisticated computer models are invaluable. Geographic information systems (GISs) are integral for such analytic modeling. They help analyze and forecast effects and convey information in an easily understood and accessible format.

Indeed, GISs are becoming an important framework for understanding a dynamic world. Users can overlay tropical storm prediction patterns from the National Weather Service on roads and other critical infrastructures to determine which areas the storm is most likely to affect. 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 which roads and businesses to close to ensure public safety.

The scope of GIS implementation is also extensive—from a desktop implementation to an enterprise operation investing significantly in data storage and processing assets. It’s not surprising, then, that a large commercial GIS industry has emerged to provide customers with software tools, training, and source data. However, the cost of large deployments using commercial software can be prohibitive. In these cases, open source applications have an equal potential for large-scale implementation across a broad range of platforms, but they can be far more cost-effective and flexible.¹

We first realized this in April 2007, when Noblis began implementing a GIS for evaluating flood inundation and network resilience. The project
required extensive storage and significant computing power on a limited budget. So, we investigated open source alternatives and eventually implemented an end-to-end process using only open source components. Although the process seemed daunting at first, the end result showed that it was well worth the effort.

**Required Components**
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, which enable queries such as “select all municipalities in 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 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. Flat files store data such as satellite images, which the system typically uses only for displaying background images to help the user’s visual orientation.

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 databases.

A central server with lots of memory and fast processors is also desirable, because processing geospatial data typically requires numerous spatial computations. Personal desktop computers running standalone GIS software are adequate for smaller applications, but enterprise-scale problems require server-based processing, memory, and storage capabilities.

Oftentimes, users reviewing results from a geospatial analysis don’t need the full suite of tools to perform the underlying geoprocessing or any background maps and datasets for geographic locations outside their area of interest. *Web-based publishing tools* bridge the gap between the 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 applications such as Google Maps or Google Earth.

Last, but by no means least, are the *visualization tools*, which can range from view-only browsers to full-function GIS desktop applications with a rich set of geospatial processing capabilities. Geospatial data is layered, 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 the 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 (for example, different symbols for points, varying line thickness, and different color polygons). Desktop clients provide this functionality in a user-friendly manner with a legend for all layers.

The ability 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.

**The Open Source Geospatial Community**
Like other open source communities, the open source geospatial community comprises hundreds of companies and individuals working toward a common goal. Support for this community comes mainly from two organizations. The
Open Source Geospatial Foundation (OSGeo) is a nonprofit organization that supports and promotes 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 working to develop publicly available interface specifications. (See the “Related Links” sidebar.)

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 standards-based libraries that enable interoperability among packages.

Figure 1 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 for all supported formats. It also comes with a variety of useful command-line utilities for data translation and processing. GDAL supports more than 50 raster formats, and OGR supports more than 20 vector formats. Other libraries—such as the Geometry Engine Open Source library—provide additional types of processing tools.

Standards
The OGC’s primary contribution is OpenGIS Specifications, which support interoperable solutions to let the Web as well as wireless and location-based servers 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 not only among open source tools but also among open source and commercial GIS software. For example, with WMS, simple browsers can view fully rendered maps, and WFS provides geospatial features that desktop GIS clients can use as layers.

Getting Started
Before an enterprise can effectively use open source tools to manipulate geospatial data, it must understand the data types it will acquire and ensure that the data formats are compatible.

Data Types
All GIS data is either raster or vector, differing in storage requirements and the operations it allows.

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—and file size.
If the raster values represent light levels, they ba-
sically present a picture of the geographic feature
of interest. The most popular variant is visual
satellite imagery; other forms include eleva-
tion data, which helps show river and ocean bed
topologies and is useful in other remote sensing
applications.

An important raster feature is that users can
mathematically combine data from multiple
rasters that cover the same area, deriving new
information, which the users can then display
as a new raster. Because raster data doesn’t
compress well, it requires vast amounts of hard
disk storage space and typically isn’t stored in
databases.

Vector data, in contrast, uses points, line seg-
ments, and closed polygons to represent a wide
range of real-world features efficiently, including
points of interest, roads, and geopolitical re-
gions. 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 exchang-
ing 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 for-
mats is suitable for images. However, geospatial
information about the raster, such as the start-
ing pixel’s coordinates, the projection used, the
pixels’ horizontal and vertical resolution, and the
standard pixel values must augment the raster
data. Some formats require embedding such in-
formation in the file itself; other formats require
a supplement file. Typical file formats for ex-
changing raster data include GeoTiff, JPEG2000,
and NetCDF.

Another set of file formats is based on XML,
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.” Most GISs—both commercial
and open source—can read and write in a range of
popular file formats. Output formats are mostly a
matter of local preference and knowing the likely
user community for the data.

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

In contrast, many major cities have 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 datasets 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.

**A Sample Open Source Environment**

After gathering the data, the next step is to find the tools to help store, manipulate, visualize, and analyze it. We successfully implemented these functions with open source components.

Figure 2 shows the Noblis GIS environment, which uses tools that map to the required components discussed earlier. Specifically, our 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;
- **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 Secure Sockets Layer (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 programming 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 us use the PostgreSQL server as a backend spatial database for GISs, much like ESRI’s Spatial...
Database Engine or Oracle’s Spatial extension. As Figure 3 shows, user clients can visualize spatial objects stored in PostGIS directly, or Grass can store the topology and use PostgreSQL as a database to store attribute data. With this arrangement, Grass can access the full range of SQL commands and functions in PostgreSQL, which it can use to manipulate data and control what’s displayed.

**Data Management and Processing**

The US Army Construction Engineering Research Laboratories (a branch of the US Army Corp of Engineers) developed Grass in 1982 for land management and environmental planning. It has since evolved into a powerful utility with broad application in many scientific research areas.

Grass has a stellar reputation for flexibly processing raster data, and version 6 added equally significant capabilities for processing vector data. Unlike most vector GIS programs, which 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—such as on-the-fly data reprojection—are lacking. However, in an enterprise architecture, programs 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 specific content. The University of Minnesota developed the program as part of its 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 acts as a service-oriented architecture for clients that need full GIS functions served over the Web. Any Web browser can access finished maps as images, so that a larger population can share the results.

**Desktop Clients**

Desktop clients generally don’t 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 reproject geographic data on the fly.

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.

Meanwhile, uDig uses Eclipse Rich Client technology to provide a complete Java solution...
Open Source

Google Earth

Google Earth provides access to high-resolution imagery from around the world, and its built-in feature layers provide a wealth of content and let users 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 them 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.

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

Although many enterprises are generally familiar with GIS software and its applicability, the idea of building GIS capabilities from open source components might seem overwhelming. With determination and the desire to explore options, open source GIS can be highly desirable. We’ve only touched on what’s available to provide large-scale performance on a small-scale budget. The tools we 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, the power of open source is in its supporting community, where user feedback drives innovation. As part of this 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.

**Reference**


**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 semitransparent ground overlay.

**David A. Garbin** is a senior fellow at Noblis. His research interests include telecommunications technology, networking, network design and optimization, economic analysis, and voice and data communications. Garbin received an MS in electrical engineering from the Massachusetts Institute of Technology. Contact him at david.garbin@noblis.org.

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