Non-Intrusive Techniques for Enhancing Decentralized Data Storage with Strategic GIS Visualization

Stephen Davies, Roger Lamb, Nicholas Odhiambo
Dept. Of Computer Science
University of Mary Washington
Fredericksburg, VA

Abstract

The benefits offered by graphic visualizations of spatial data are difficult to reap when that data is physically distributed throughout an organization. If rank-and-file employees maintain addresses of clients on their personal workstations, for instance, it is impossible for strategic decision-makers to analyze this data en masse*. Converting the entire organization to a centralized system would of course enable the strategic analysis. But this involves a number of drawbacks: devoting hours of employee time to training on a new system; costly and possibly error-prone data migration; and the disruption of the workflow already in place for individual workers.*

In this paper, we present a completely open-source GIS solution deployed in actual practice that offers spatial visualization of distributed address data without disturbing current business practices. We describe how this approach can be adapted to other settings, and give practical tips for practitioners.

1. Motivation

The advantages offered by graphic visualizations of spatial data are becoming more widely appreciated as GIS systems become more prevalent. Organizations that maintain street addresses of customers, for instance, would in many cases be well-served to have an up-to-date digitized map of the locations of those customers, annotated with key discriminating information, for policy makers to analyze.

Many of these organizations, however, have existing procedures in place that may preclude the wholesale introduction of a new software system. For example, a consulting firm might employ dozens of consultants, each of whom maintains lists of clients and prospective clients in spreadsheets on their desktop computer. Migrating all this data into a new system with a centralized data repository would indeed make possible a strategic visualization of all the firm's clients, but at a considerable cost. Hours of employee time would have to be devoted to training on the new system. The organization's existing procedures – which are familiar and may work well for employees – would have to be overhauled. Any customized information used only by a particular worker would likely be lost as they are forced to use a generic system common to all. And the migration of the data itself would be manual and painstaking.

In these circumstances, the most desirable solution would be one in which the employees' tools and techniques could remain unchanged, and yet information pertinent to a high-level analysis – including strategic geographic information – could be automatically extracted and made available to policy-making staff. In this paper, we outline a complete technical solution for realizing these goals, based entirely on open source software.

2. Case study

The Spotsylvania County Parole Office in Fredericksburg, Virginia is a small, public agency that exhibits the features described above. Paroled convicts (offenders) are each assigned to one of about two dozen parole officers (POs) within the organization. Each PO maintains an Excel spreadsheet located on their own desktop machine to track information about the parolees to whom they have been assigned. The structure of these spreadsheets is rather simple: one row corresponds to one offender. The columns in each row include the offender's current residential address, plus standard fields like name, type of crime, the date parole was granted, etc. Also included are fields that impact the PO's schedule of activities, such as the next scheduled home visit for each offender, and the date by which a urine test is expected. A given PO might have as few as thirty offenders assigned to them at any one time, but some have over 200.

In some cases, a particular PO will also maintain information customized to their particular purposes. One spreadsheet might contain a column for nearby landmarks to assist in locating residences in the field,

for example; another might include a record of each offender's "supervision level" which mandates a specific frequency of contact. In this way, the form of each spreadsheet varies somewhat. Historically, each of the spreadsheets evolved from a common source (when a new PO is hired, they typically inherit a copy of a previous employee's spreadsheet to modify and use) which lends a high degree of commonality to them all. There are differences, however, since each PO is a unique individual who finds varying kinds of information useful in performing their duties.

The organization's daily information usage, therefore, is highly decentralized. Workers are productive and content using flat files on their desktops to satisfy their daily information needs. Indeed, from a tactical standpoint, the agency's information management technology is a success.

From a policy-planning perspective, however, this scheme has severe limitations. The police chief and other strategic planners have no way to analyze the trends and patterns in this data that is distributed across a myriad of different systems. They may be interested, for example, in identifying which areas of the county have high concentrations of sex offenders, so they may be singled out for special monitoring. They may wish to see time-series data on offender relocations. They may find it necessary to more aggressively monitor the behaviors of offenders who live at the geographical fringes of certain regions. They may desire to redistribute PO responsibilities so as to minimize travel. None of this is possible with the current data management technology.

3. A non-intrusive solution

An attractive solution to this problem can be achieved by leaving the spreadsheets and the existing procedures in place, and extracting the data at periodic intervals to populate a centralized database. The architecture for this solution is presented in Figure 1. Custom code written specifically for this solution is depicted by white boxes; all other boxes indicate freely-available open source tools.

The system is comprised of the following components:

- **Satellite extractor programs.** A simple Java program is installed on each employee's workstation, along with the Java Virtual

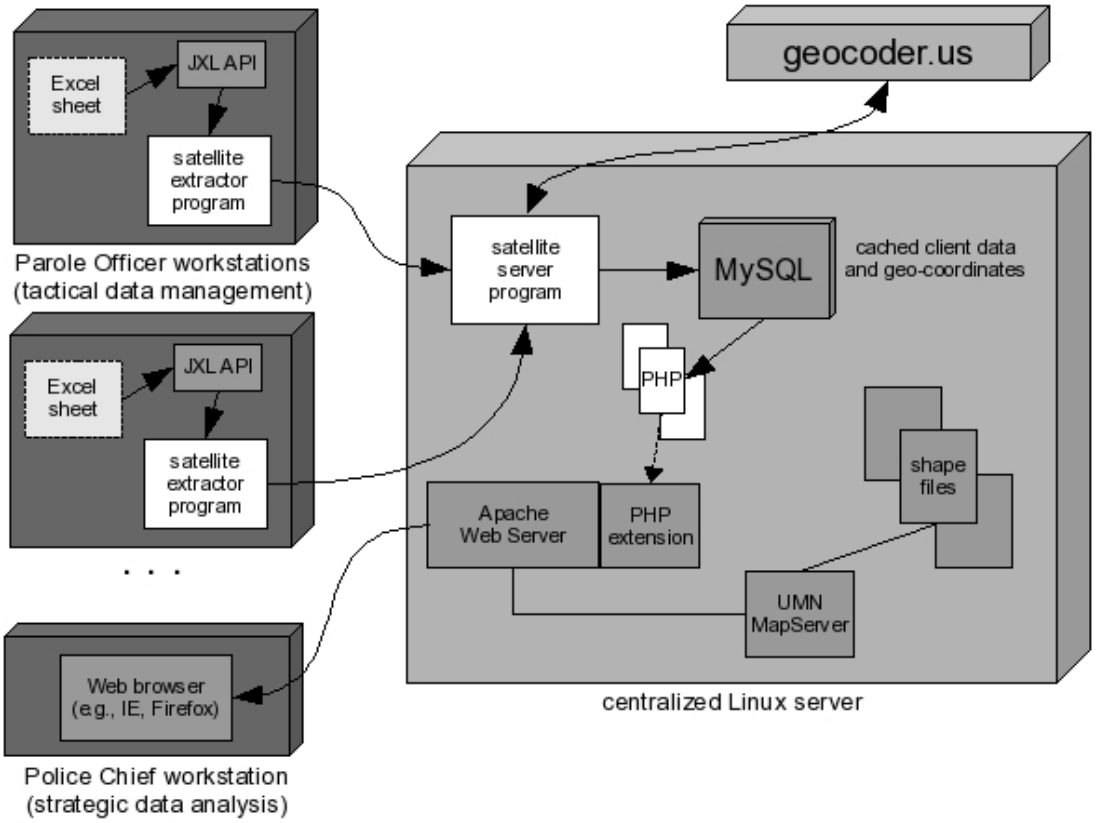

Figure 1. Solution architecture. White boxes represent custom code written for this implementation; medium gray boxes represent open source software, data, and services.

Machine[1] required to execute it. This program uses the open source JExcel API[2] (packaged and deployed in a single .jar file, or *Java archive*) to access the spreadsheet's data and process it a row at a time. JExcel provides classes and methods to specify filesystem locations of Excel spreadsheets, and then iterate through the rows and columns, etc. The satellite extractor program makes certain assumptions about the spreadsheet format – namely, that the relevant data appears in the file's first sheet, that it consists of a single table, that the first non-whitespace row of the sheet contains column headings, and that these column headings will include (among others) two mandatory fields, called "ID" and "address." Other than these constraints, the program makes no other assumptions (there are no limits on the number of rows or columns, for instance, nor what the non-mandatory columns can be named.) Information from the entire table is sent, one row's worth at a time, to the satellite server program which answers to a known, open port on the Linux server. The extractor programs are configured via the standard Microsoft Windows "scheduled tasks" facility to run at regular intervals, in our case once per day.

- **Centralized Linux server.** The solution requires a dedicated machine to house cached data retrieved from clients and to serve Web pages for data analysis.
- **Satellite server program.** This Java program runs continuously on the Linux server, listening on a known port for input from satellite extractor programs. When data from an extractor program is received, it writes it to a MySQL database, also deployed on the server. Thus the database contains a cached version of the total contents of the client spreadsheets, out of date by at most one (configurable) scheduled interval.
- **Free geocoding Web service.** As the satellite server program writes each parolee's residential address to the database, it checks to see whether the database already contains geocoded longitude/latitude coordinates for that address. If not, it invokes the freely-available Web service[3] geocoder.us[4] to obtain these coordinates. If so, it presumes that the coordinates previously stored are accurate. Note that if a given residential address were ever to change geographic locations (as could happen on rare occasions if new construction or city planning initiatives reconfigure street configurations), the long/lat coordinates in the database would be permanently out of date, resulting in an inaccurate data point plotted on the map. This exceedingly rare occurrence could be dealt with in a number of ways, perhaps best by periodically flushing the long/lat information from the database and forcing a fresh geocoding of all addresses (at the cost of some delay.)
- **MySQL database.** The database of cached information has an extremely simple, generalized schema, essentially storing the contents of each spreadsheet cell as a key/value pair. (The key of each pair is comprised of a parolee ID and spreadsheet column heading, and the value is simply the contents of the corresponding cell.) This allows individual employees to create whatever spreadsheet columns they choose, and to name them at will, without concern that database schema changes would be required to stay in sync with spreadsheet structure. As noted above, the database also contains long/lat coordinates for each address. (This is for the sake of map generation performance, since invoking the Web service for each address every time an analyst accesses the digitized map is prohibitive.)
- **Apache Web Server with PHP extensions.** The digitized map and associated functions are generated by server-side PHP code executing under the open-source Apache Web server.
- **PHP pages.** The data extracted from the spreadsheets and cached in the server's database are spatially rendered by a set of custom PHP pages that Apache executes. These pages call MapServer (see below) to actually generate the map. The pages also provide scaling, panning, and filtering functions on the data. They also display the non-geographic data as desired, in the form of a popup window that appears when the analyst hovers over a displayed geographic point (see Figure 2.)
- **MapServer.** MapServer[5] is an open source platform for generating Web-based interactive spatial mapping applications, originally developed at the University of Minnesota. Architecturally, it is a CGI program that Apache invokes on demand to generate map images. This occurs when a strategic analyst accesses the main PHP page in the application. The PHP page requests that MapServer generate a map with the appropriate pan and zoom settings, and render annotations for each parolee's geographic location. The geographic information for the actual map data are provided by files in ESRI's shape file format[6], which can be obtained for any U.S. county from the Census Bureau[7].
- **Analyst workstation.** Finally, a police chief or other strategic analyst can access the map at any time from an ordinary Web browser. This provides a consolidated, graphical, mostly up-

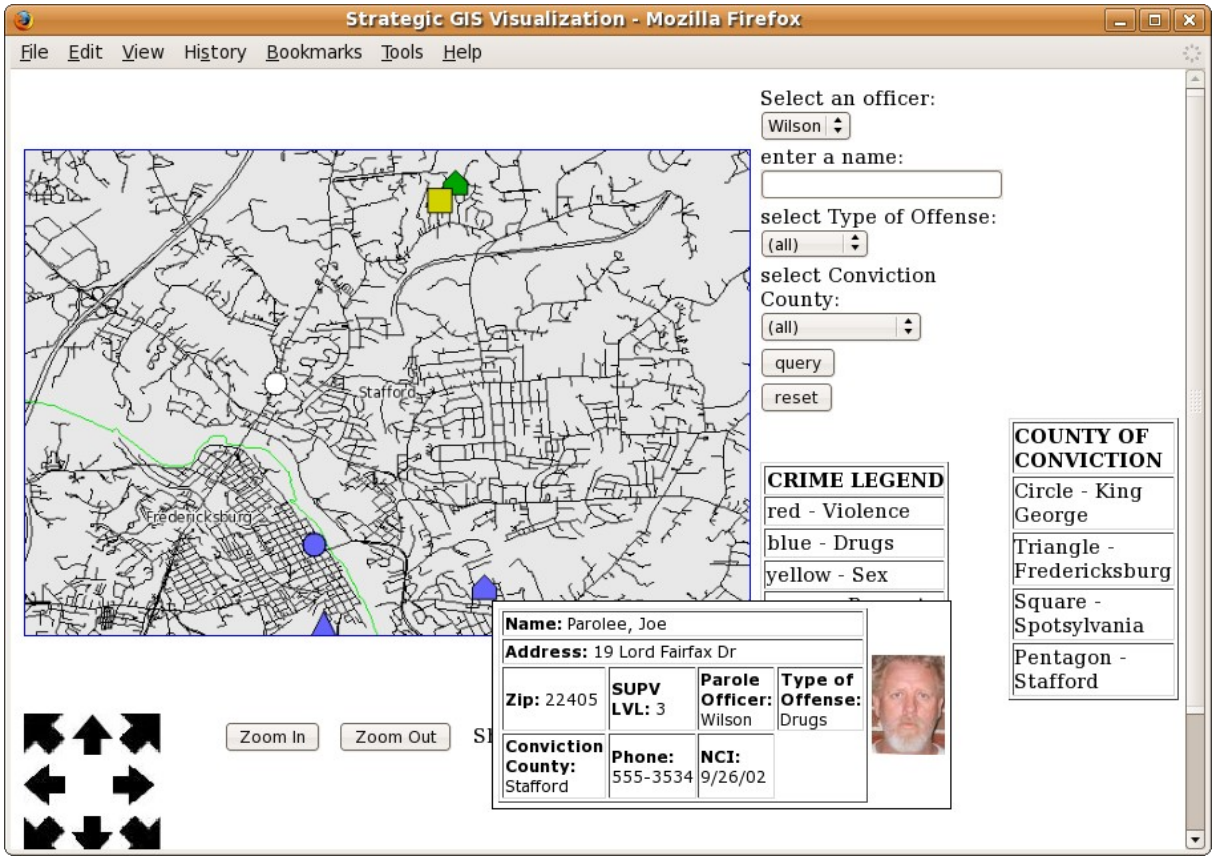

Figure 2. Web-based GIS visualization application. Analysts can pan, zoom, and filter the data based on selected criteria. Hovering over any geocoded data point brings up a pop-up menu with the coded fields that correspond to the columns on the spreadsheet for that parolee.

to-date view of the contents of the distributed spreadsheets at any one time. (As stated above, it is understood that the data presented may be out of date by at most one satellite extractor interval. This interval, however, can be set as small as desired, at the modest cost of more frequent bursts of network traffic.)

4. Impact and suitability

Put simply, the impact of the system is that a strategic-level graphical view of the entire organization's day-to-day information is made available to analysts, without the rank-and-file employees having to modify their existing procedures at all.

This brings three major benefits. The first is that the information is accessible at *all*: the data, until now scattered across a multitude of employee machines in diverse filesystem locations and formats, is suddenly integrated and made manifest to managerial staff via a consolidated interface. Secondly, this data is filterable along a number of different dimensions, as shown in Figure 2, giving the analyst flexible control over what subset of the data they choose to view. And thirdly, of course, the GIS mapping takes simple tabular address information and turns it into a pannable, scalable, richly detailed two-dimensional map that is immeasurably more likely to illuminate geographical patterns.

It is important to realize, however, the specific circumstances that make this kind of solution attractive. Specifically, it lends itself to settings in which:

- ✔ employees currently manage their activities using files on their local machines, and this approach works well for them;
- ✔ there is a great deal of commonality to the structure of the files different employees use (in our case, spreadsheets that originated from a common source and were modified within certain limits);
- ✔ either the expense of purchasing an enterprise system, and/or the disruption caused by

migrating workers to new tools and procedures, is perceived to be too costly; and

- ✔ the strategic data analysts are content with a read-only view of the data, and one which is understood to be "mostly up to date" at all times.

In these scenarios, the kind of solution outlined in this paper can bring great benefits to an organization's information management, at very modest cost.

5. Limitations

It is worth pointing out a few limitations that go with this approach:

- **Client machine format constraints.** Once this solution is in place, employees are thereafter somewhat constrained in how they can modify their own local files. Altering the structure or information in these files beyond the limits of what the satellite extractor can handle may result in lost or incorrect data. Hence this needs to be communicated to employees, and could prove burdensome if employees find these constraints frustrating.
- **Satellite extractor deployment.** The solution requires another (small) program to be installed on client machines, which means it must be maintained, and re-installed on any new workstation an employee might upgrade to.
- **Bandwidth.** Depending on the number of employee machines, and the size of the data on each, the network traffic generated by the periodic extraction and upload may become significant. Ways to minimize this problem include increasing the extraction interval, of course, and/or staggering the extraction times of the client machines so that they do not all contact the server simultaneously.
- **Stale analytical data.** No matter how short the extraction interval is set, there will still be *some* time during which analysts will see data on the website that is stale with respect to recent updates on client machines. This is not normally expected to be a problem, as the OLAP paradigm typically presupposes an analysis of offline, archived data from an OLTP system.
- **Out of sync geocoding.** As explained above, for performance reasons our system does not request geocoding for addresses that have already been previously geocoded. The assumption is that residential addresses are associated with physical dwellings that will not change longitude or latitude. In cases where this is violated, the only option that will prevent stale coordinates is a periodic flushing of the database cache.

6. Discussion

Finally, note that this approach aims at a point architecturally midway between today's unfortunate status quo (an organization with data residing on scattered, heterogeneous systems) and a fully-featured, centralized, enterprise GIS system which is the target of many research and development efforts (see discussion, e.g., in [8],[9].) We believe that in some cases, this kind of non-intrusive solution can obviate some of the difficulties described in [10] and avoid some of the organizational changes prescribed in [11]. It is appropriate for organizations who are largely satisfied with a decentralized data management approach, and desire only to add the ability to view this data strategically.

7. Implementation notes

Notes on implementing this solution, including the installation and use of JExcel, MapServer, MySQL, shape files, Apache, PHP, and geocoder.us, is available online at http://paprika.umw.edu/~stephen /nonIntrusive.html.

8. References

[1] Sun Microsystems. *Java Platform, Standard Edition, version 1.5.* Available at http://java.sun.com/javase.

[2] *The Java Excel API, version 2.4.* Available at http://jexcelapi.sourceforge.net/.

[3] The World Wide Web Consortium. *Web Services Activity Statement.* Available at http://www.w3.org/2002/ ws/Activity.

[4] Locative Media Lab. *geocoder.us.* Available at: http://geocoder.us.

[5] University of Minnesota. *MapServer version 5.2.1*. Available at: http://mapserver.osgeo.org.

[6] Environmental Systems Research Institute. *ESRI Shapefile Technical Description whitepaper.* July 1998.

[7] U.S. Census Bureau. *2008 TIGER/Line Shapefiles.* Available at: http://www.census.gov/geo/www/tiger/ tgrshp2008/tgrshp2008.html.

[8] Busser, D. "The evolution to enterprise GIS." *ArcNews Online*, Winter 2007/2008.

[9] Keating, G.N., P.M. Rich, and M.S. Witkowski "Challenges for enterprise GIS." *URISA Journal*, **15**(2), pp. 23-36. 2003.

[10] Nedovic-Budic, Z., and J.K. Pinto, 2000, "Information sharing in an interorganizational GIS environment." *Environment and Planning B: Planning and Design* **27**(3), pp. 455 – 474. 2000.

[11] Oppman, R.S., An Organizational Model for Successful Enterprise GIS. In von Meyer, N.R. and R.S. Oppman (Eds.), *Enterprise GIS*, (Park Ridge, IL: URISA), pp. 44-52. 1999.