Integrating GPS Data within Embedded Internet GIS

Arunas STOCKUS Alain BOUJU Frédéric BERTRAND Patrice BOURSIER
University of La Rochelle - L3i
Avenue Marillac
17042 La Rochelle - FRANCE
Tel. +33 (0)5 46 45 82 62
{arunas.stockus, …}@univ-lr.fr

ABSTRACT

In this paper we investigate the development of an embedded and mobile geographic information system. Its main characteristics concern the possibility to access various information sources and to provide the basic functionalities of a navigation system, e.g. positioning.

This system uses a differential Global Positioning System (GPS) device to acquire the position of a mobile (e.g. vehicle). A cellular phone with an Internet based connection permits to access distant data sources and to transfer data between the components of the system. A Web browser and a Java applet are used for data integration and visualization.

As it is used in the context of a mobile application, the basic software component in the architecture of our system is a local embedded server. It ensures real time access to local GPS data, but also to multimedia and spatially referenced data which are stored on distant servers.

We also present some results of practical experiments that have been carried out with this system embedded in a vehicle.

Keywords

Embedded GIS, Mobile System, GPS, Spatial Data Integration.

1. INTRODUCTION

An embedded geographic information system (GIS) is a specialized system that could mostly be considered as a mobile navigation system. Its typical characteristics concern the presentation of geographical information to the driver, positioning and guidance. However, due to their reduced computational power and their autonomy, many actual navigation systems have some limitations. The geographical information they can use covers only limited areas, and its precision may be insufficient in some situations. The autonomy of the system complicates data updates and access to distant information sources, especially those providing dynamic information. The specific technologies used for communication between a mobile system and a data server require the installation of specific infrastructures and limit the area of its usability.

But the actual development of technologies changes the way information can be accessed and processed. It minimizes the differences between desktop applications and embedded ones by bringing the power of desktop computational technologies into mobile devices. It gives the possibility to use the same kind of applications and data access modes with both kinds of environments: presentation of multimedia information, connection to the Internet, etc.

In this paper we present an embedded GIS, which is based on such technologies. Its architecture is a client-server one, and the system components may give access in real time to distant information servers. We use a satellite based localization system (GPS) for the geographical positioning of a mobile unit (e.g. a vehicle). The connection to distant data servers at the physical level is based on the use of a cellular phone. At the software level, we use an Internet connection for exchanging data between the components of the system. Information integration, visualization and processing is made possible by using a Java enabled Web browser and a Java applet.

We introduce a local embedded server in the architecture of our system. Its main goal is to ensure optimal real time access to different data sources: (i) local ones, for example data sent by the GPS device, and (ii) distant ones, for example data stored into a spatial database which is located on a distant server.

In the following section, we present the framework of our study, i.e. the requirements for an embedded GIS and the possible technological solutions. We also present the choices that we have made and the main problem they lead to: access and integration to data located on various sources, and especially those sent by the GPS device. We then review various solutions, and we present the solution that we have implemented. We describe the architecture of the system and the principle of its functioning. We then present some remarks coming from tests results. We finally review future developments of ongoing work.

2. FRAMEWORK

We consider an embedded geographical information system (GIS) as a mobile system that helps the user to locate himself. Such a system makes possible the presentation of spatially referenced and associated multimedia information. It is also a positioning and guidance system that gives a user the possibility to locate himself on a map.

As a consequence, such a system must be able to access and to integrate in real time information of different types and precision. It can be static (e.g. maps) or dynamic information (e.g. vehicle

ACM GIS '99 11/99 Kansas City, MO USA
© 1999 ACM 1-58113-235-2/99/0011 … $5.00

position) that must be integrated continuously. The information may be provided by different sources: local data sent by positioning devices (e.g. GPS), or data accessed from distant information servers, for example traffic conditions. Spatially referenced and multimedia information can also be embedded locally or stored on distant servers and transmitted at the request of the user.

So, the access to distant data sources, especially those providing geographical data, and the integration of data in real time are important characteristics of an embedded GIS.

2.1 Access to Distant Geographical Data

The use of the Internet and a Web browser for accessing, processing and visualizing spatial information is one of the aspects that actually give strong interest in the field of research and development of GIS. We distinguish two techniques which are used to access spatial information through the Internet: server-side oriented methods and client-side ones, depending on which side data processing and calculations are done.

The use of Common Gateway Interface (CGI) scripts [Gun96] is a typical example of the first method. This is a "classic" means of interaction between a browser and a Web server. A form of an HTML page is used to collect information and to send it to the server, where a special program – a CGI script – processes it and sends back an answer (the generated HTML page). In the case of spatial data, we usually collect the parameters for a spatial query, send them to the server and receive the HTML page containing a map image.

This method has some drawbacks. The calculations are basically made on the server, and the client only visualizes the result. It implies a high server load and intensive communications with clients. Besides, it is impossible to establish a permanent connection between the client and the server for a work session. This complicates the reuse of calculation results and the responses to previous queries.

One solution to this problem, as well as its application for the processing of geographical data, is presented in [12]. CGI scripts are used to perform the calculations, and intermediate results are stored on the server. "Cookies" (small packets of information memorized by the browser) are used to simulate a permanent work session by navigating trough HTML pages. Thus the results of queries that were executed previously can be re-used.

Plug-ins, Java applets and ActiveX components are the main tools used with client-side oriented methods.

A plug-in is a software component, which processes a particular type of data. A browser uses it each time it receives an HTML page containing references to such data. A plug-in offers more flexibility on the client side than CGI scripts. As modules containing an executable code, they cater for local calculations and for a more advanced exchange of data with a server (permanent connections, progressive data loading, etc.)

Autodesk MapGuide [1] is an example of a plug-in, which makes it possible to visualize spatial data within a browser. It also allows the evaluation of queries locally and on the server, the loading of different groups of data according to the needs of the users.

A drawback of plug-ins is that the component must be installed on a browser before it can be used. Thus, it is dependent both on the browser and on the operating system, and it has to be developed for each platform.

As is the case for plug-ins, Java applets and ActiveX modules are software components executed by a browser or by an operating system. The difference lies in the fact that the corresponding code is loaded from a server and does not have to be installed previously. It can be used on any platform and by any Java or ActiveX enabled browser (the latter is currently limited to Microsoft Windows operating system).

[5] introduces a system prototype called GeoLens which visible part to the user is a Java applet. It accesses data by means of various exchange formats. The prototype itself is based on the OpenGIS specification [3]. A similar approach is used in the prototype Lava [6] which uses an Internet connection to access geographical data and a Java applet for its processing and visualization, as well as in [10] where the GAEA Java applet is presented.

[9] presents a software component called GeoLib, which uses an ActiveX module for the visualization of vector-based spatial data.

In all that cases, software components get geographical data "directly" from a Web server ([10]) or by some middleware (special purpose servers) used in conjunction with that one ([9], [6], [1]).

2.2 Our Solution

The embedded system that we present in this paper is also based on the use of the Internet to access geographical and multimedia data from distant servers. It uses a Web browser and a Java applet for data presentation and processing. As we said before, we consider it as a mobile navigation system connected to distant data sources. We use a differential GPS device to obtain the position of the mobile. Communication between the embedded system and distant servers is based on the use of a cellular phone. These technological choices allow us to have a system easily workable and accessible. It does not need heavy investments, so much for its implementation as for its use.

Indeed, current GPS devices, and especially differential GPS, have a good precision which enables positioning on the map in real time. Mobile communication networks are widely practicable, and they now cover very large parts of the territory, especially in urban areas. Web browsers are now common tools for the presentation of almost all kinds of multimedia data. They also become the execution environment of more and more sophisticated applications. Finally, they can be considered as indispensable tools, found on almost all computers and allowing to reach information anywhere on the Internet.

The Java language allows to develop small portable applications ("applets") that can be easily and safely downloaded from distant servers, and then executed by Web browsers on various operating systems. Applets can download the required data and process them locally, without having to reconnect to the server and send a query. It reduces the cost of connection. Applet's code can also be loaded from a server at each new work session, which ensures easy updating of applications.

As we will se in the next section, the use of Web browsers and Java applets for data visualization and processing introduces some difficulties concerning data access. But it has some advantages too, because the field of use of a system based on such

technologies is very wide and not limited to mobile and embedded applications. It can also be used to access geographical and associated multimedia data everywhere we can connect to the Internet : desktop workstations connected to local networks, home computers connected to the Internet by a telephone line, etc.

2.3 Problematics

The development tools that we have chosen are already available, and they can easily be used by developers. But some difficult problems still remain, namely : (i) the diversity and increasing number of information sources, (ii) the diversity of information types, especially for spatial data, (iii) the low speed of communications between the components of the system, and (iv) the diversity and increasing number of users.

Information can be in different forms (geographical maps, images, video or audio data) and in different formats (for example, raster or vector images in the case of maps). With spatial data, we also have to cope with differences in the precision of data, projection systems, visualization parameters, etc.

Another aspect linked to the diversity of information is the diversity of its sources, whether situated on distant servers or stored locally. One example of this aspect would be GPS data, acquired and then transmitted by a device connected to the computer, and having to be displayed on a map loaded from a server.

The use of communications based on cellular phones seriously limits the speed of data transfer. Moreover, multimedia information tends to occupy large amounts of memory. It may happen that the usual means of transferring information on the Internet (e.g. using Hypertext Transfer Protocol (HTTP) or File Transfer Protocol (FTP)) are no longer acceptable.

In the following sections, we present a prototype of an embedded system which must cope with two main problems : (i) access to information acquired by a GPS device, and (ii) integration of GPS data and map data in real time.

The first problem is due to the diversity of information sources having to be processed in a mobile unit, and to security restrictions imposed to Java applets executed within a Web browser [7]. Indeed, an applet executed by a Web browser has to gather information coming from a distant data server. At the same time, the applet has to receive positioning information sent by the GPS connected to the computer, which correspond to local resources. As these belong to the physical resources of a computer (for example, connected to a serial port), they can only be retrieved and processed by the native code of the operating system.

Because of security restriction imposed on an applet, it can only gather data coming from the same source as itself. Thus it cannot access to local resources that include the file system and peripherals connected to the computer. A method must be designed to communicate data from a GPS to an applet.

The second problem is due to the difference of information precision and the different projection systems used with maps and GPS. Several transformations of GPS data have to be done to integrate them with maps.

3. ACCESSING GPS DATA

We have considered several methods to communicate the GPS data to the Java applet. The applet should then process them and integrate them with other data, possibly downloaded from distant servers.

3.1 Principle

One solution would be to create a specific application (program executed by the local operating system) that can access GPS data and send them to the server. In this case, the applet can receive data from the server, the same source as its code.

The problem with this solution is that the server should cope with many more connections : those of the applications processing GPS data, and those of the applet requesting it. Also, in the case of connection interruption between the client and the server, the applet could receive no data from the GPS, even if they were available on the client.

Another solution would be to use an electronically signed (“trusted”) applet. The navigator can no longer impose security restrictions on such an applet. In this way, applets can access directly various data sources, including local ones. But this solution has also some drawbacks. Access to the GPS is made by the native code of the operating system. So the applet has to be able to check if it has already been executed on the client, analyze its execution environment, load and install all necessary software components, etc.

Experience gained during our first developments has shown that the applet code should be as simple as possible, if we want to run it on different browsers and operating systems without making any prior modifications. Access to the physical resources of a computer would make it very dependent on the operating system.

The solution that we have adopted derives from the first one. We use a local mini-server installed in the computer (an embedded server). The principle of this solution is that the applet accesses all the necessary data via this server (see figure 1). The server redirects the applet’s connections to the other servers or access local data. Then it communicates the answer to the applet. In this case the local server is the single source of data for the applet (and the Web browser). The applet does not have to violate security restrictions by connecting to various data sources, because the local server takes care of it.

A disadvantage of this solution is that the user has to think to install supplementary software in order to ensure the good functioning of the system (as in the first solution). But there are also some very important advantages.

The embedded server can take care of communications between the server part and the client part of the system and hide it from its other components. This allows us to consider some methods of connection optimization, at the same time keeping the same mode of communication with the other components. For example, the local server can use the HTTP protocol to exchange data with the browser and the applet, while some specific protocol should be used to exchange them with distant servers. The possibility of optimizing communications presents a considerable interest in the context of an embedded system, as this system limits the speed of data transfer.

3.2 Organization of the System

The organization of the system is shown in figure 1. A Java enabled Web browser, a data visualization applet and an embedded server are the software components on the client side. The browser and the applet are the components used for processing and visualizing the information. The embedded server acts as an intermediary between the browser or the applet on one side, and the "whole world" on the other side.

HTML pages containing the references of the applet's code and the applet code itself are data available on the server. It is accessed firstly and it corresponds to the HTTP requests sent by the Web browser or the applet.

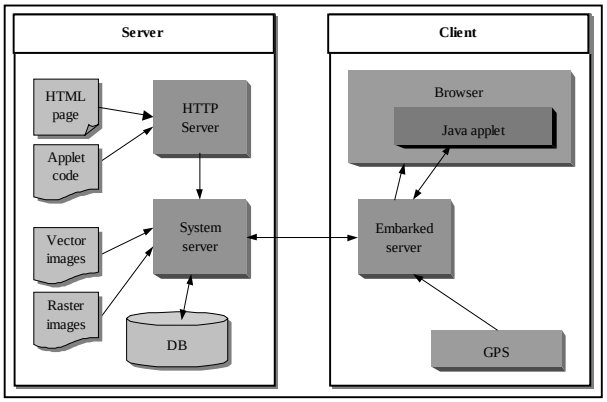

Figure 1: Architecture of the system

Other data transmitted from a server to a client are stored in databases or file system. They are processed by the applet. It can be geographical data (vector-based and/or images) as well as associated alphanumeric data. It might also be information of practical nature, such as the number of free places in a car park.

The software components on the server side are an HTTP server and a specific system server. The system server acts as the entry point for all clients, and it processes only data requests (geographical, alphanumeric or multimedia data). It redirects the HTTP requests towards the HTTP server. There are several reasons for separating the tasks between the two servers.

We "reuse" the code of the HTTP server and ensure the correct processing of HTTP requests. A system server can in turn process the data requests more efficiently. It can connect directly to a database (the Web server has to use CGI scripts in this case). It can process the data of a map locally and send it in parts according to the needs of the client (the Web server can only send whole files). The system server increases data security, because there is no direct access to the databases, or to the image files from the client. Finally, some communication optimization can be used between the embedded server and the system server.

3.3 Working principle

The system's working principle is given in figure 1 (the arrows show the data flows).

The Web browser connects to the embedded server and loads an HTML page and the applet's code. Once loaded, the applet starts its execution, connects to the embedded server, loads and visualizes the required data, i.e. raster images and/or vector images making up a map. At the request of the user, the applet sends a query to the embedded server to obtain GPS data, receives the reply, processes it by making all necessary transformations and visualizes the position of the vehicle on the map. The applet can also obtain the positions of vehicles stored in the database.

In turn, the embedded server works on the client computer. It can receive requests from the applet or the Web browser. After receiving a request, it connects to the distant server, sends the request, receives the answer and sends it back to the caller. It processes the request locally if access to GPS is needed. It sends to the applet the GPS data recovered by a serial port. It can also send GPS data to the system server so that they can be stored in the database and reused by other system components.

The operating principle of the system server is similar to the one of the embedded server, and it depends on the type of request. It can establish a connection to the HTTP server and send back the answer (HTML page, applet code or other data) to a client. The system server can read and send raster or vector-based data stored within files. It can also make a connection to a database for processing a query and send back the answer to a client. A query can be a request for updating data or a simple selection query.

Thus, we solve the problem of GPS data access from a Java applet by introducing an embedded server. GPS data integration with geographical data is done by the applet, which transforms it into the projection system of the map.

4. TESTS AND RESULTS

4.1 Implementation

All parts of the system are implemented as Java applications. Consequently, they are portable applications and they can be executed on any operating system supporting the Java Runtime Environment.

The GPS being connected to the serial port, the embedded server accesses it using a *javax* package, extension of the Java Development Toolkit. The system server establishes a connection to a database using JDBC (Java Database Connectivity) functions. Moreover, the system server is independent from the actual HTTP server, since it communicates with it only by means of HTTP queries.

The visible part of the client is a Java applet (figure 2). It allows loading and visualization of data that make up the different layers of a map, i.e. vector-based layers and raster images. These layers constitute static information that do not change during a work session and do not need to be updated, once they have been downloaded. A map can also contain dynamic layers that represent the position of the vehicle(s). In this case, the applet regularly sends queries to the embedded server to obtain new GPS data. After receiving the answer, the applet converts GPS data within the map projection system and visualizes them. The applet behavior is the same in the case of a connection to a database.

The applet thus allows visualizing of different information layers: raster layers, vector layers, and also the position of the vehicle. It also allows basic operations on maps displayed on the client: scale change, zooming and panning.

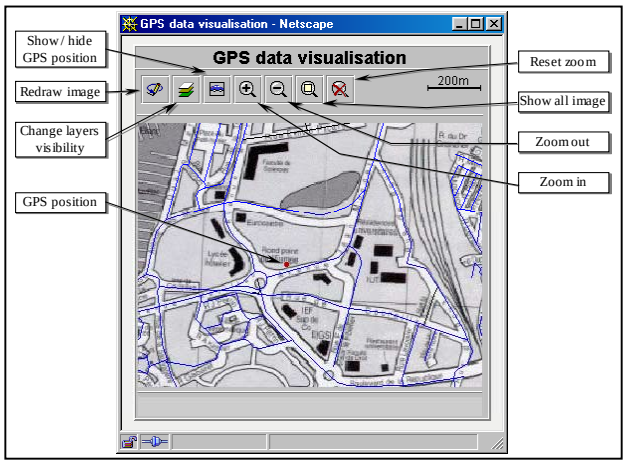

Figure 2: A Java applet running on the client

We have also implemented in the embedded server a function that simulates the operation of the GPS. It works on GPS data which have been recorded during tests, conducted under real use conditions. It allows to test the operation of system components when the GPS is not available.

4.2 Tests

We have tested our system under different configurations and different platforms. For the mobile tests, we have used a portable computer with Microsoft Windows 95 operating system along with a connected GPS device and a cellular phone. The last one was used to make connection with distant data servers. Geographical data were transferred from the server to the embedded client by establishing Internet connections. GPS data acquired during the tests were sent to the server as User Datagram Protocol (UDP) packets and then stored into a Postgres database.

Workstations with Windows NT, HP-UX, Sun Solaris and Linux operation systems connected to a local network along with simulation of the GPS were used for non-mobile tests.

Figures 3 and 4 show samples of GPS positions acquired during our (real mobile) tests. Figure 4 presents GPS data (in bold) compared to a map. Figure 5 shows GPS data acquired during two different tests.

As we can see, data precision is the main problem with geographical and GPS data integration [11]. It can have different causes: the imprecision of measurements of the GPS and the geographical data or the errors of data transformation between projection systems.

Figure 3: A vehicle trajectory indicated by the GPS.

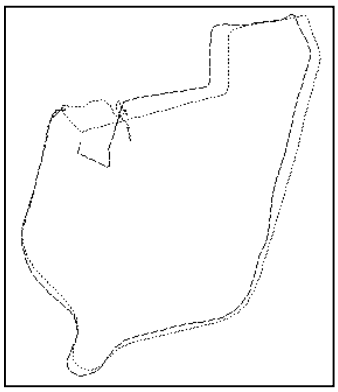

Figure 4 : Two trajectories indicated by the GPS.

We have used for our tests maps in the Lambert II Carto projection system (which is standard for maps produced in France). Data sent by the GPS were in the WGS84 projection system and have been converted into Lambert II.

We also encountered some difficulties for connecting to distant data sources. In fact, connection with a cellular phone is too slow and not very reliable for transferring large amounts of multimedia and other data in real time. However, its speed is quite acceptable for exchanging small packets of information, like the vehicle position, traffic conditions, some reduced maps or some parts of more detailed maps.

5. FUTURE WORK

The optimization of communications between servers and clients is one objective of our future work and experimentation. We anticipate two main directions:

- The use of a data cache, which consists in doing a local copy of the data that have been downloaded by the applet. It will relieve us from having to reload them at the beginning of the following work session. We should also use some data stored on a local disk or CD-ROM with those loaded from a server. In all cases, the synchronization of data must be implemented, as it was done in [2].
- The definition of new modes and protocols for communication between servers, in order to reduce the amount of information to be transmitted. Such an optimization can be based on data compression and on its serialization. In the first case, we reduce the size of information to be transmitted. In the second case, we reduce information itself by selecting only some parts of it.

Another direction of ongoing work concerns the integration of map data and GPS data. A simple transformation of data between different projection systems is not sufficient because of precision. Map matching methods must be applied to find the exact location on the map (a point on a street) to which the GPS position corresponds. Some of them are presented in [2]. They permit the integration of the GPS position into a vector-based map and use the calculation of distance between geographical objects : points, lines, curves.

Finally, we plan to study different organizations of the system in order to adapt it to the user needs in various contexts : mobile computer, desktop computer, computer with no GPS data available, etc. The actual architecture of the system offers a large flexibility, and many configurations are possible :

- The removal of the local server. If we do not need to access local information sources, and if the communication speed is sufficient, then we can operate such a modification. For example, it can be the case for accessing distant data servers from a desktop computer connected to the local network;
- The removal of the system server. This situation is similar to the previous one, as the local server can take care of some of the functionalities of the system server (for example the connection to distant databases);

The removal of both servers. This case is the simplest and it permits to have a lightweight system, because the Java applet is the only part to be used. It accesses data directly through an HTTP server.

6. CONCLUSION

The embedded geographical information system that we have presented in this paper is based on very popular and widely accessible technologies, namely an Internet based connection for accessing various data sources, and a Web browser with a Java applet for its processing and visualization. They become universal means of information diffusion and presentation, especially for multimedia information.

Such technologies are usual for desktop applications. But our first development and our tests show that they can also be used in the framework of an embedded and mobile application. It gives several advantages because it makes differences between desktop based and embedded application disappear. From one side, it facilitates the reuse and adaptation of existing applications and solutions. From another side, some solutions to crucial problems of mobile application can be adopted for non-mobile ones too. An example could be the optimization of communication based on mobile connection. The same principles of optimization can also be adopted in the case of "usual" Internet connections, which often suffer from limited data transfer speed.

Finally, it gives the possibility to develop and reuse some components of the system without taking care of the actual context of its utilization. In our case, an example is the spatially referenced information presentation part, i.e. the Java applet. Without any supplementary modification, this component is used as part of the system developed in the framework of the *Magic Tour Net* project. This project is an evolution of *Magic Tour* [4], the ESPRIT Project 8752, concluded in 1997. Its major emphasis was devoted to the development of a multimedia authoring system specifically oriented towards the development of tourism-oriented applications. Magic Tour Net extends Magic Tour by adding the possibility to develop such applications through the Internet. The applet that allows the visualization of spatially referenced information is used as one of the multimedia information presentation tools.

7. ACKNOWLEDGMENTS

The research which is described in this paper has been funded partly by the region Poitou-Charentes and the city of La Rochelle.

8. REFERENCES

[1] Autodesk MapGuide : State-of-the-art network-centric GIS application architecture for publishing and accessing geodata, 1997. Autodesk White Paper. http://www.autodesk.com/products/whtpaper/

[2] D. Bernstein and A. Kornhauser. Map Matching for Personal Navigation Assistants. Transportation Research Board, 77th Annual Meeting, Washington, January, 1998.

[3] K. Buehler and L. McKee. Introduction to Interoperable Geoprocessing and the OpenGIS Specification. Open GIS Consortium Technical Committee, Third Edition, June 3, 1998.

[4] P. Boursier, D. Kvedarauskas, S Iris and A. Guilloteau. MAGIC TOUR : the Integration of Multimedia and Geographic Information Technologies in an Authoring System. In Spatial Multimedia and Virtual Reality, Research Monographs in GIS, Eds. A. Camara and J. Raper, Taylor & Francis, 1998.

[5] C. Behrens, L. Shklar, C. Basu, N. Yeager, and E. Au. The Geospatial Interoperability Problem : Lessons Learned from Building the GoeLens Prototype. 1st Int. Conf. on Interoperating Geographic Information Systems, Santa Barbara, CA, December 1997.

[6] C. Berg, F. Tuijnman, T. Vijlbrief, C. Meijer, H. Uitermark, and P. Oosterom. Multi-server Internet GIS: Standardization and Practical Experiences. 1st Int. Conf. on Interoperating Geographic Information Systems, Santa Barbara, CA, December 1997.

[7] J. S. Fritzinger and M. Mueller. Java Security. Sun Microsystems, Inc., Java White Papers, 1996. http://www.javasoft.com/docs/white/index.html

[8] S. Gundavaram. CGI programming on the world wide web, O'Reilly, 1996.

[9] D. Kvedarauskas, P. Boursier, X. Culos, T. Deltheil, and S. Iris. GEOLIB : A Software Component for Making GIS Tools Interoperable. 1st Int. Conf. on Interoperating Geographic Information Systems, Santa Barbara, CA, December 1997.

[10] D. Kotzinos and P. Prastacos. GAEA, a Java-based Map Applet. 1st Int. Conf. on Telegeoprocessing, Lyon, France, May 1999.

[11] V. Noronha. Towards ITS Interoperability – Database Error and Rectification. Int. Workshop on GIS-T and ITS, Chinese University of Hong-Kong, April, 1999.

[12] M. Szmurlo and J. Madelaine. A Network of Asynchronous Micro-Servers as a Framework for Server Development. 6th Int. World Wide Web Conf., Santa Barbara, CA, April 1997.