Proceedings of TeleGeo'2000: Second International Workshop on Telegeoprocessing. 2000, p. 81-91.

Java Application Architectures to Facilitate Public Access to Large Remote Sensed and Vector Geographic Data[1]

P. Fernández, J. Nogueras, O. Cantán, J. Zarazaga, P.R. Muro-Medrano[2]

Computer Science and Systems Engineering Department
University of Zaragoza
María de Luna 3
50015 Zaragoza, SPAIN

pedrofb@ebro.cps.unizar.es
jnog@ebro.cps.unizar.es
ocantan@ebro.cps.unizar.es
javy@posta.unizar.es
prmuro@posta.unizar.es
{http://iaaa.cps.unizar.es

Abstract: This paper shows some practical advantages of the use of Java in development of GIS visualisation software. The authors propose and develop three Java application architectures to solve a common GIS problem like accessing and managing large image data, from the point of view of different users and net requirements (local, intranet and internet). The paper is also an example of how it is possible to implement the three architectures, reusing most of the application code, and offering users a portable system able to coordinate the management of large image data through different points of the net.

Keywords: *Java, GIS, large image data, applet, web server, information system, distributed Java architectures, object orientation*

Introduction

The Java programming language has gained quick and large acceptance as a prominent programming language in recent years [SM98]. Its programming capabilities, multiplatform implementations, facilities for developing distributed applications with powerful built in high level mechanisms of communication, its powerful standard libraries and the extensive availability of public domain software, has become Java as one of the most suitable tool for software development.

In our days Internet and the web are, with no doubt, the prefer medium for sharing information and one of the most powerful mechanisms to provide access to public information where user interactivity is required in some way. In addition to the features explained above, one of the reasons for the Java impact in the programming community has been its ability to integrate in the web.

The GIS community is living an impressive increment of interest for the java-web technologies. In this sense, the most relevant GIS and spatial data management commercial companies are devoting significant resources to develop software products with internet facilities, much of them written or with the ability to interoperate with Java (Esri, MapInfo, Oracle, Intergraph, …), the reader can refer to [CLH99] or [LIMP99] for product reviews and comparisons. Dozens of applets with some GIS functionality can be found in the internet, many of them publicly available (see [SM98] for a review with interesting web links). We can mention systems like Descartes [AAC99], GAEA [KP99], GeoToolKit [BBBCS99], inovaGIS [GC99], …, which where presented in the previous TeleGeo'99 conference in Lyon. On the other hand, efforts of standard organizations have been displaced to the Java and internet worlds (the OpenGIS Consortium has prioritized Java Implementation and the Web Map Server Interface Specification [OPENGIS99] against other feature and service standards).

[1] **Acknowledges**: This work was partially supported by the *Comisión Interministerial de Ciencia y Tecnología* (CICYT) of Spain through the project TIC98-0587 and by the *Consejo Superior de Investigación y Desarrollo* of Aragón through the project P-18/96.

[2] To whom correspondence must be addressed.

Let us explain a prototypical real example where this kind of technology can be applied. Two years ago, the Spanish Environment Ministry decided to purchase satellite images (from SPOT and Landsat) covering one of its zones of interest: the Hydrographic Basin of the Ebro River. This region has one of the most significant rivers of Spain, both in caudal and length. The images of the basin should help geographers to demarcate rivers, water reservoirs and, of vital importance, irrigated lands and crops. Satellite images cover more than 85.000 Km^2. In order to improve social profit of this investment, the administration decided to expand the access to this information within the Ministry personal, as well as the general public, which may be interested.

To increase the usefulness of that information for the user and easier the access, the Ministry decided to provide the required software for visualisation. The software must have the operational features of traditional GIS visualisation systems and the ability to mix the images with other vector data with at least some minimum accessibility and speed requirements. Offering the users a similar look of traditional GIS systems could help geographers to quickly find different geographic accidents over the images, taking profit of the large image repository managed next to the superimposition of vector coverages. Problems arise in dealing with large images that must be managed and sent thought data communication nets and the different strategies for application development (local, intranet and internet applications, every one with its own speed and size restrictions) each one with its programming effort exigencies.

To get the needed software we check some commercial products and realize that, normally the products had high license charges and programming constraints and, on the other had, we had to purchase different products for each of the application strategies. To avoid that, a decision was made to development our own GIS visualisation components to manipulate the imagery and the vector data and Java was chosen to implement the three suggested strategies. Java allow to distribute the developed GIS component along the net, from server to client, and allowed us to build three different architectures adapted to different users, from a local one with all data integrated, to Intranet and Internet architectures with a remote repository of large data.

The decision to use Java appears now as essential. We had to develop different application architectures for different requirements but we where able to reuse most of the software code in the three and the internet clients where tremendously easier because the use of Java. We also benefited from the availability of Java specialized standard classes to work with images. The objective of this paper is to describe this technical experience. The rest of the paper has been structured as follows. First of all the application environment and its related problems are described. Section 3 deals with the problems involved in managing large images whereas the section 4 describes four different architectures, based on Java, that provide solutions for different system and user requirements. The paper ends with a conclusions section.

Application context

Our objective is to develop a technology capable to facilitate the access to an image repository from different connection places. The type of access to information from a user, local, through an Intranet, or in the general case, via Internet will establish the kind of operations supported, and the speed and the content of the transmission.

The same problem appears from different points of view. The main requirement, public access to large and common data, must be solved, but also is necessary to combine the different ways of transmission requirements. Before building anything, we have to analyse the technologies and architectures, to avoid the building of too much independent systems suitable to each kind of deployment platform.

Java has been selected as programming language. This is a multiplatform language that allows the use in any platform having a Java interpreter. This characteristic, besides its facilities to develop distributed software, the process interconnection management, and the support of the language by traditional Internet browsers have influenced this decision. Another additional characteristics like its image management libraries have helped in the development of the application

All the architectures proposed have the same main module, the Visual GIS, in charge of location, load and display of the images and vector data coming from the built repository. The module is extended in each case with the necessary modules to comply the new platform function requirements. The local architecture consists only of this module. Local data does not make necessary any special data access.

Before confronting the software building process, we must pay attention at certain constraints imposed by all these architectures, overall in access time and speed of transmission. The large data repository force us to get any good solution about management of images. Firstly, we could think in a simple solution: Why do not we build a whole image of our interested zone and work with it?. This solution would decrease greatly the complexity of the application, as we could work just using a traditional GIS system like ArcView. However, users would not be very glad, as they see that an operation, like a simple zoom, might last minutes. Mainly, this problem is due to the big size of that image, but thinking in our previous requirements and if we add a transmission medium like Intranet, or in the worst case Internet, the limited band-width will deteriorate even more the answer rate.

Previous solution forgets a basic point, users only need a little piece of the total image at one moment of time. The proposed solution is to cut the source images into others small ones, performing an image repository, which can be used by an specific application. Not only the repository will be composed of images with the same pixel scale than the original virtual photography, but we also will add images with greater covered area per pixel, in order to speed up the visualisation of greater zones than covered by the small images. [BGS99]

All the architectures proposed have the same basic module, in charge of location, load and display of the images and vector data coming from the built repository. We have made an effort to achieve a versatile module with the basic GIS characteristics, initially only ready to work in a local environment. The VisualGIS module is thought to be improved, with a few additional work, to expand its use through Intranet and Internet. Clearly the several ways of deployment require the existence of some additional modules specialised in each platform type. Besides, some hard operations, like insertions, will be only present in the local or Intranet version. Each type of architecture suggest also to move the computational cost through the net; with Internet we need a powerful server, but the compute in Intranet can be local, with just a simple layer and image server into the main server exactly.

Intranet architecture introduces new complexity not covered by the basic module. Data will be stored on a server, being accessible from all the clients. At the server part, an http web server provides the clients with the image and vector data through the http protocol. A new server process is activated, which access to attribute data of vector coverages, using Java Database access protocol (JDBC). The client is composed of two modules, the mentioned VisualGIS, and as an extension, the modified access methods to data through the data server. Java is a net-oriented language that facilitates greatly the data transmissions between processes, so the required modifications are minimum.

The application code does not need to reside on the client. Java applet mechanism allows to download and execute directly the code with a browser, like Netscape or I.Explorer. In order to speed up the initial access time, the same code can be also executed on the client with the Java interpreter

In the Internet architecture, oriented to home users, the same display module is used, but in this case, it is executed on the server. One top module is added over the VisualGIS which must recover the internet map requests and transform them into zone selections, that the VisualGIS module can execute. After this is generated, a JPEG image and an html page are sent to the client as answer.

ARCHITECTURE	CLIENT	SERVER
LOCAL	VISUALGIS + DATA	X
INTRANET	VISUALGIS+ DATA CLIENT	DATA SERVER + DATA
INTERNET	HTML + JPG	SERVLET + VISUALGIS + DATA

Fig. 1. Module composition of Local, Intranet and Internet architectures

Dealing with sets of large images

The following points describe the processes of splitting and composition applied to source images, necessary to conform the image repository, and the required steps for any application wanting to find and recover images from the repository.

Image repository generation

Source image files are organised making an irregular grid, made up 217 files. Each file is a georeferenced SPOT image in BIP format of about 30x20Km and 5 meters/pixel of scale. The images on the grid are overlapped, so each image is bigger than its grid ideal square. The grid is stored in the Shapefile format, of ESRI.

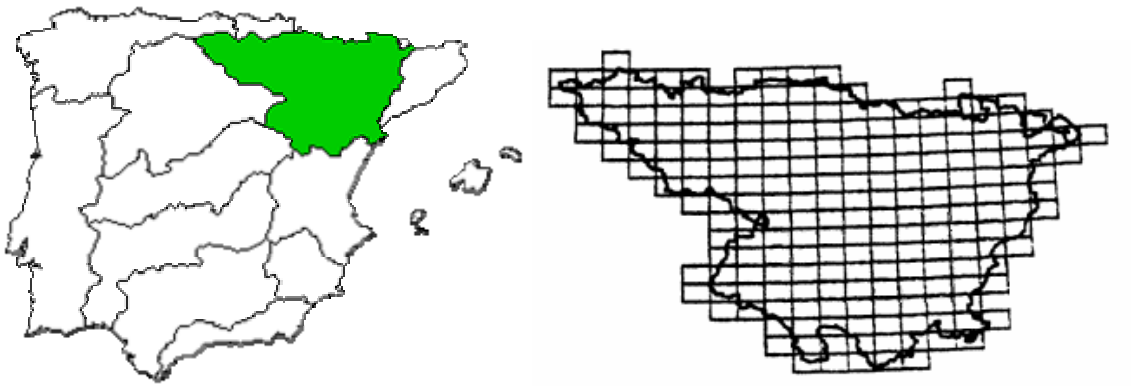

Fig. 2. Hidrographic Basin of Ebro location and its grid splitting in SPOT images

Each image is around 6000x4000 pixels, and takes up over 80Mb of disk space. The BIP format only stores RGB points, without any type of compression. Some other graphic formats like JPEG, GIF or PNG allow to reduce image size without loss of quality (or minimum). To reduce the total size of images (21 Gb!!!) we can use any of these formats; In our case, we have selected JPEG, for its optimum compression rate, and because JAVA v1.2 has standard methods to comprise raster images to JPEG files.

In order to speed up the access to images, images compression is not enough. We have to cut the original ones into small jpeg images so as to ease transference and load. The resulting image repository will be composed of lots of small images organised in different scale levels. All the images of an scale level make up a virtual image of the basin. The first level is a whole image of the basin, and the last one is composed of images at the same scale level as original photos. One image of a top level covers four images of a down level.

The building of the image repository involves two processes. The first process is to cut the source images, for composing the down-levels. The second process is to combine the images in order to create top scale levels. Both processes are guided through ideal grid structure. These processes create a seven scale level image repository. The fourth level, and also medium scale level, match exactly with the size of the grid rectangle. This size, at 1:50000 scale, is the normal scale of work with images in the ministry. The next level, the fifth, divides exactly the previous level in four zones, obtaining images at 1:25000 scale. In the same way are constructed 2 down levels more, until seventh. The reverse process allows building top levels.

To implement these processes we could have selected a manual way of work, using ArcView for example, but if we calculate the final number of images making a simple multiplication:

TOP Levels: 1+ 217/(4x4) + 217/4
MEDIUM Level: 217
DOWN Levels: 217x4 + 217x4x4 + 217x4x4x4
Total number of images ≅ 18600

We can see that the manual process is not suitable. Therefore, we have developed two little software programs in Java that perform the two tasks automatically. The first program, the Cutter, takes as input all the photos and the grid information and generates fourth to seventh levels, one source image at the same time. The next one, the Composer, groups images of fourth level and creates the three top levels. Larger levels (5-7) are not processed by the Composer as the decrease of quality is not appreciated in top levels(1-3) .

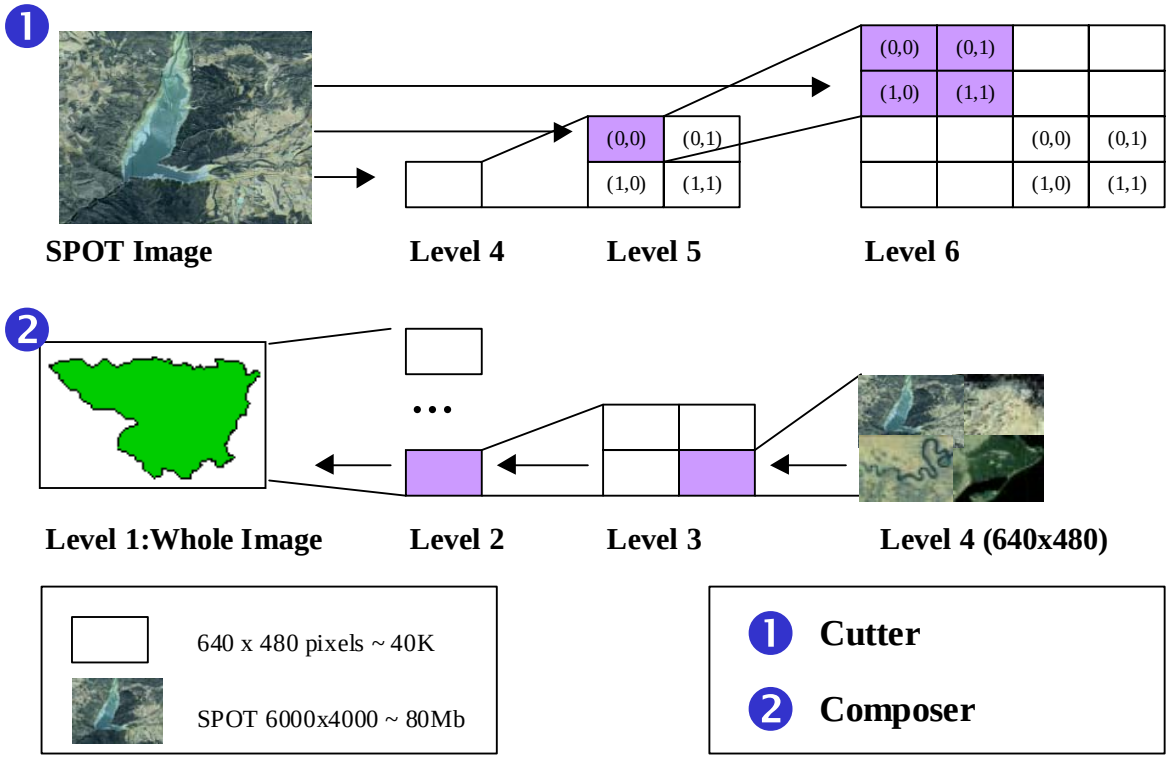

Fig. 3. Generation process of the image repository

At the end, the whole repository, composed for compressed jpeg images have a total size of 1 Gb. Each jpeg image has a dimension of 640x480 pixels and its size approximately rounds 40K. The compression factor obtained by the use of jpeg format is 1:25. The total size of the image repository is not affected too much by the image duplication method, obtaining a final size ratio of 1:20(1Gb:21Gb) slightly higher than jpeg compression ratio, however it improves substantially the user access time to top zones. This low increment of the total size is owing to the quadratic decrease of the image number from a level to the next one. The initial seventh level has 18600 images, the sixth level have 4750 images, and the fifth counts only with 1200 images approximately.

Image recovery

Images are stored using the structure of the grid. Each image is located inside the grid through its row and column number. For example image *5x12* will be stored in a file called *h5_12.jpg*.(*h* from Hidrographic). Down levels use a similar nomenclature. The four parts of a level are called with relative column an row numbers. The North West one is the image *0x0* and the South East image is named *1x1*, so in the previous example, image *h5_12.jpg* in level four, match with images *h5_12_0_0.jpg, h5_12_0_1.jpg, h5_12_1_0.jpg, h5_12_1_1.jpg* in level five. The next down levels follow the same criterion.

A top level groups previous levels in quartets. One image of a top level is named with the same image number as its corresponding top-left image of the next down level, and a prefix showing what level belongs to the image. Following the example, image *hl3_4_12.jpg* belongs to level 3 and is composed of images *h4_12.jpg, h4_13.jpg, h5_12.jpg and h5_13.jpg. We must mention that* in level three all the image numbers are divisible by 2, in level two all by 4, and in the first level by 8. This makes possible an easier search.

The first step to locate the images which form a geographic zone is to find the appropriate level of scale. The best level will be the one which covers the zone with the minimum number of little images. The following piece of code shows the algorithm to calculate the level of an image zone:

```
ViewDimension = Dimension (width, height) of the selected view zone

LevelDimension1 = GridSquareDimension x 2 x 2 x 2

LevelDimension(n) = LevelDimension1 / 2^n
```

```
n = 7

while ( (ViewDimension > LevelDimension(n)) || (n >= 1) )

   n = n - 1;

end while

ViewLevel = n;
```

Fig. 4. Top level algorithm for detection of the suitable image scale level

A parallel step to the location algorithm is the detection of the specific grid rectangles which contain the four corner points of the view zone. This operation is done using the search mechanism offered by the general visualisation module over the vector layer which contains the grid.

With the corner locations and the scale level, and using the size relations of images from different levels, is possible to detect the images of the selected level which contains the corner points. Finally it is only necessary to find the files which correspond to the images inside the rectangle defined by the four found corner images.

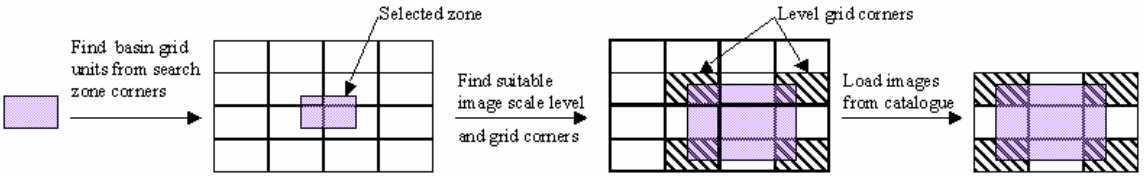

Fig. 5. Image search steps

Java application architectures

In this chapter we are going to describe the entire development process of the system. It exposes the three steps of the software development, from the initial local application to the final html page. In the last point, we talk about the future of the Internet application, in accordance with OpenGIS Web Map Server Specification [OPENGIS99]

Thick client. Isolated application

We wanted our software to follow an evolutionary cycle of life. First, it was needed to develop a software able to view and manage such vector data as the generated images, all of them in the local machine. Once this software was ready, the next step involved the improvement of the area of use, through Intranet. And the last objective of the project, making profit of the developed software, should be to develop a web map server with the available information. This server, accessed by web pages, could be visited by home users.

The main application is composed of two main modules implemented in Java: the ImageSearch module, in charge of obtaining the composing images of a view zone, and the GISEngine module, which will be able to display image and vector layers. Java programming language was selected because it makes easier web application development.

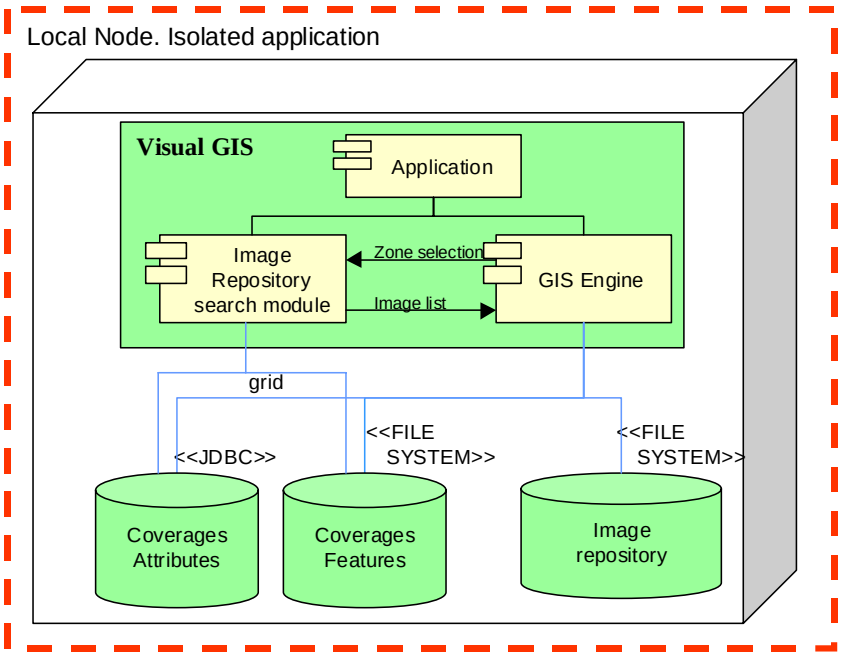

Fig. 6. Local architecture

The GISEngine module has arisen from the last years increment of GIS software necessities in the geography market, caused by the increase of computer abilities. Nowadays modern information systems include software in charge of maintaining geographic elements. The module developed offers some useful tools to manage information systems. It displays georeferenced image and vector data, and implements some of the most basic and common map tools, like zoom, pan or selections. The component also includes a basic GUI (Graphic User Interface) that allows other applications to integrate the module without having to implement some common windows.

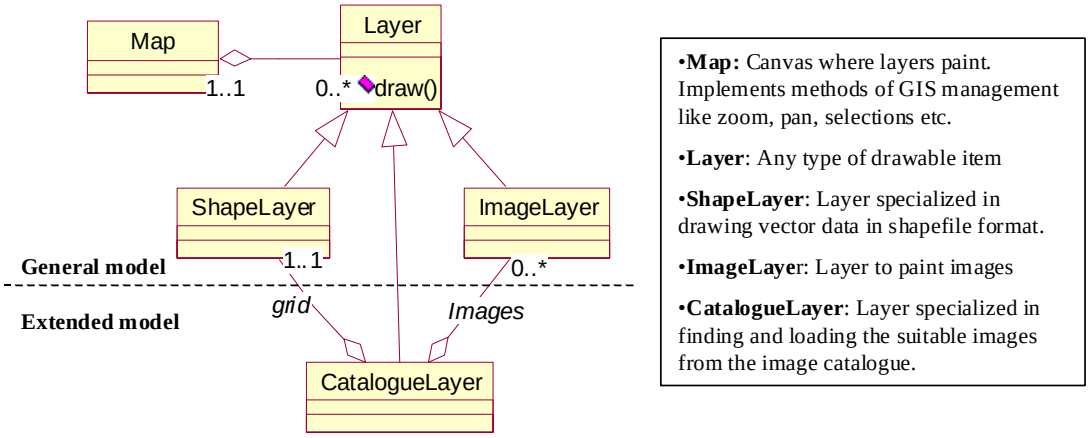

Fig. 7. Main object diagram of the GISEngine module.

The GISEngine is developed in JAVA. It follows an object oriented design, able to be easily increased. As a result of the project, some functions have been improved, and even other ones have been added, such as map saving. The development of a generic viewing support, cofinanced by several projects of this type [FERNANDEZ99], has resulted to be fundamental in order to ease the development of new GIS applications as well as in the final price. For example, MapObjects from ESRI [HART97] is a good engine of map visualisation, perhaps even the suitable one for our project, but the existence of several related projects using the same basic GIS tools, have made possible the recovery of the initial investment and the exemption of runtime licenses.

The remaining module is in charge of work with the images. Given a zone selection, with a source coordinates and an scale, it must locate and obtain those images from the repository that make up the selected zone. The RepositorySearch engine is a local component, integrated with GIS application. It does not need an intermediate

server to get image names, but of course, it is necessary to know the image naming rules as well as the location of the repository.

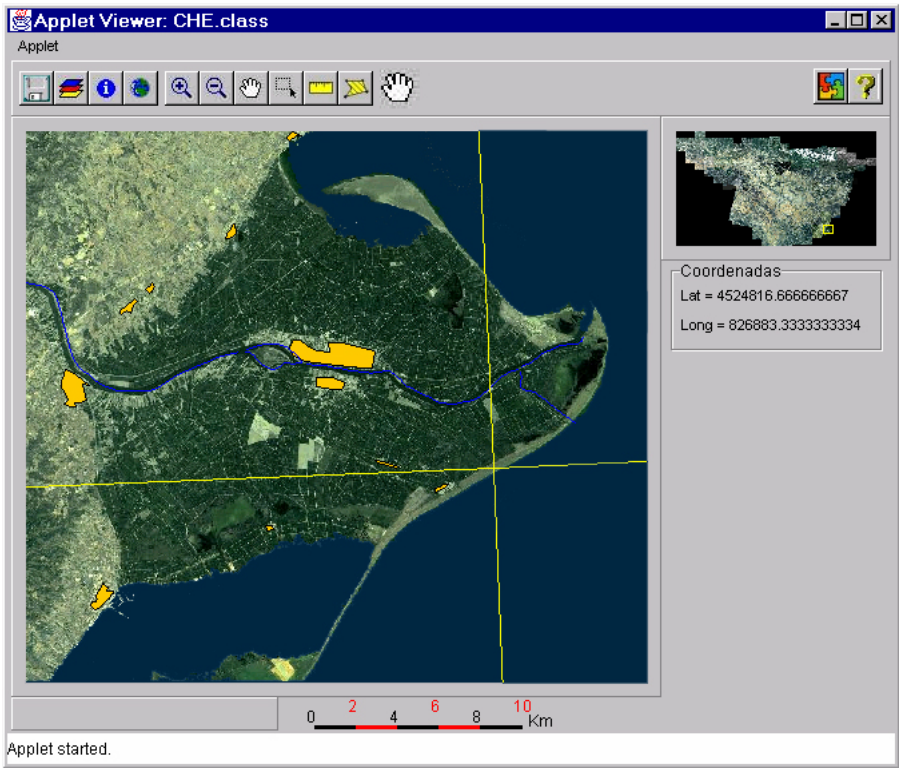

Fig. 8. General view of the application. The centre of the image shows the Ebro delta with to coverages of rivers and population groups. The top part contains the main toolbar, and on the right a general map of the basin displays the selected zone. Under it, the actual mouse coordinates (latitude and longitude) are displayed. Over the map a yellow cross represents the cutting point of four 1:50000 quadrant

Intranet access: Thick client with remote data, the Java applet

The application has been developed as a Java Applet. An applet is a mini-application module which can be launched from a browser through an html page. This way of software development allows the coexistence of local and distributed applications, without any distinction except only from the implementation of the data access methods to shared resources.

Of course, our shared resources are the images of the repository and the vector data layers. Our next step is to improve the software so as to support distributed data access. As it can be seen, the main functionality of the software will not change, only the parts related with the access to the centralised data. All the software computing tasks will go on lying on the client machine.

Application and image servers can be represented by any usual Web Server, like Apache, JWS, Netscape or our choice, IIS from Microsoft. Application software will be downloaded in the traditional way, using the html&java mechanism. The images will be located into a public directory accessible from the clients. Each client has, as in local version, a properties file indicating where to find the images.

The Java source code does not need to be changed in order to comply with download requirements: Java supports it directly. Java multi-platform even guarantee right execution over different machines or operative systems. However image securing methods should be slightly modified to support remote data access instead of local access. Java file control methods can be easily replaced with these ones specialised in remote data.

Work with vector data is a bit more complex. In the local version, to decrease the use of memory, vector layers are not loaded in memory; all the work is done against physical storage. Nevertheless, using an HTTP server, this solution it is not possible because the vector layers must be loaded firstly in memory, and then they can be displayed several times on the applet. Therefore, vector attribute data is not loaded previously at all, like in local architecture. When attributes of a feature are required, as result of a selection operation, a database query is generated against the vector database (made up by the dbase files of shapefiles) in the server part. The database server is just a simple translator of JDBC database queries from client to server, through Java Remote Interface (RMI) protocol. The client code is not affected with this change, because Java database access protocol (JDBC) is fully independent of the database source, only is necessary to change the driver. The new JDBC driver has been obtained from Gnu, and is called *RmiJdbcDriver*. The server part of the driver is also distributed by Gnu, free of payment. [GNU97]

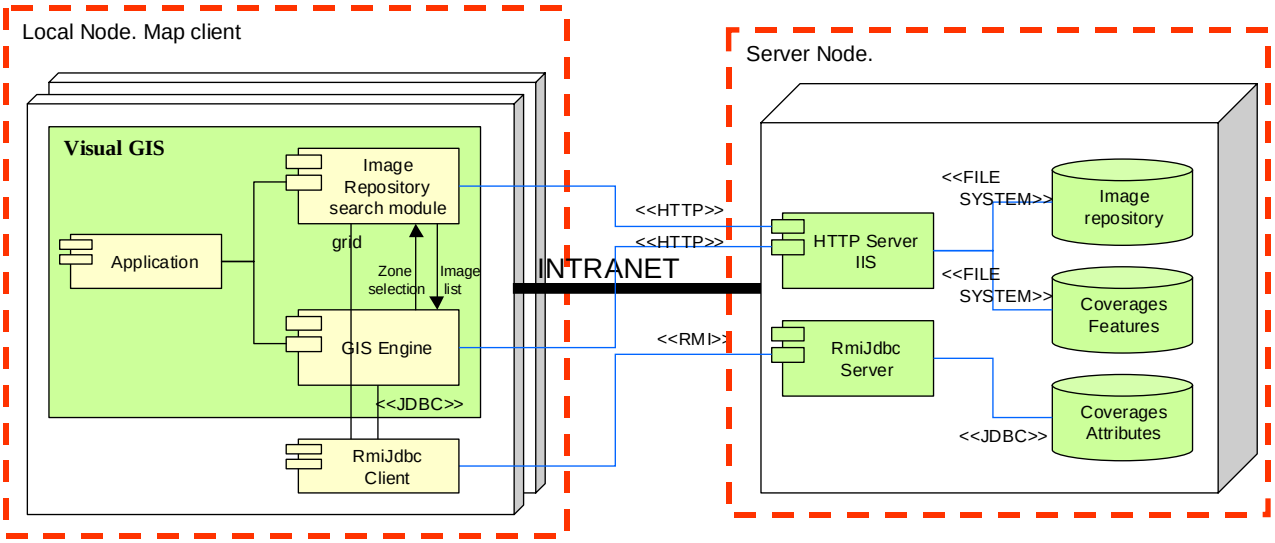

Fig. 9. Intranet architecture

This architecture allows a distributed application to use shared data with just a simple and slightly overloaded server. Clients also obtain benefits, because no local storage is required, and any modification on data is immediately known. One of the requirements of the Intranet use is relatively high speed and not too much load in order to provide clients with minimum speed access conditions. Inside the Environment Ministry, testing seat of the software, has been proven a really good running, comparable to the local version with the images stores in a CD.

Internet access: Thin client and remote image server

After the completion of the main application to be used into the Intranet of the Environmental Ministry, it is time to evaluate the benefits we could obtain from this project. Although we have invested significant and economic resources, we remain still unknown to the general public. One of the main policies of the ministry, and in general public administration, is to provide everybody with all the information it owns.

Therefore, it seems clear, that the following step will be the use of Internet to provide automatic diffusion of information. This way, we can make profit of the previous investment to generate a low cost product for public purposes. The only thing we need is a simple HTML page, with information about the vector and image data, requiring not computational cost at the client.

The solution is to transfer all the software computational cost to the server. A powerful server must be able to support lots of client requests (although not too many expected at present) and serve html pages with a simple map interface and usual operations like zooms or pans. The objective of this software is simply to show users some satellite photos with geographic accident data like towns or rivers.

Additionally, a new module in charge of client request serialisation is required. This module asks the GIS software to site over the desired zone, and returns to the client an answer page image composed of the saved view image and other additional data. This module, named MapServlet, is integrated next to VisualGis software and uses all its functions.

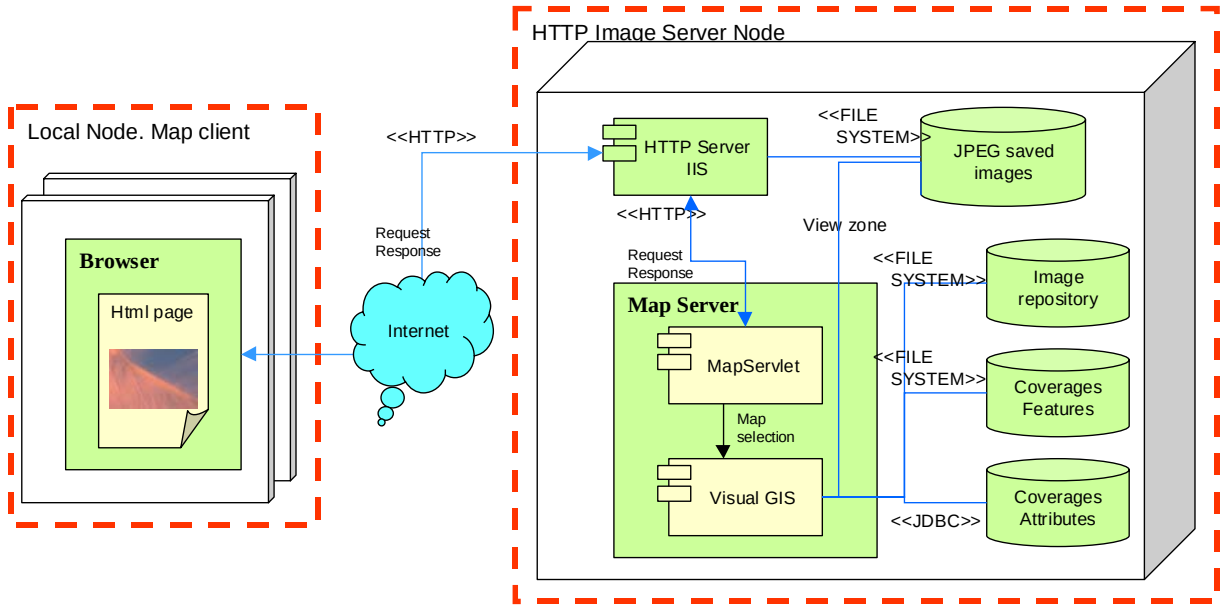

Fig. 10. Internet architecture

The MapServlet module works sequentially. First it gets the page location parameters and the executed operation, necessary to demarcate the new viewing zone. Then the visualisation module is in charge of take position over the zone in the usual form. Next the servlet invokes on the VisualGis module the Saving method, that will obtain a jpeg file containing the viewing zone. Finally the servlet builds the page using the jpeg file. Once the answer page has been sent, the server is able to process a new map request.

Generalizing internet access: OpenGIS map server architecture

As it is recognised by GIS experts, OpenGIS Consortium has been for the last years the most relevant institution in the definition of geographic software structure and data interchange mechanisms. One of our plans to expand our 'ad hoc' application in the future is to integrate OpenGIS results inside a more ambitious idea about GIS data servers in Internet. This new server would be based on the OpenGIS Web Map Server Interface [OPENGIS99]

A data web server, according to OpenGIS definition, must be able to distribute several kinds of information, like images, vector data or attributes, in different formats and coordinates. One of the ways to develop this kind of software is to integrate different applications into a more complex software, but still versatile enough to be used by other applications of the same style.

Using the interface provided by OpenGIS and implementing it with the technology developed in the GISEngine module, we can achieve a general Web Map Server module able to public GIS information on web. The interface MapServer of OpenGIS does not affect too much to the design of the Internet architecture of a Web Map Server like the one showed in previous point. The greatest improvement of this design consists in the integration of all the concepts forming a full of sense system, which provides the methods to group and interchange information.

Conclusions

This paper has shown the advantages in the use of Java to create low cost applications for GIS visualisation. Although different application architectures have been developed for different user and net requirements (local, intranet and internet), we where able to reuse most of the software code in the three, whereas the internet clients where tremendously easier because the use of Java. We also benefited from the availability of Java specialized standard classes to work with images which we used to manage large image. The Java communication capabilities allowed to build a portable set of modules capable to coordinate and access to large image data through different points of the net.

The Intranet application is currently in full operation in the ministry, and it is being accessed by its whole staff through typical web browsers like Netscape and Internet Explorer. It accomplished all requirements about access speed, and offers users a full set of management tools. The application simplifies and speed up the access to the images, only available previously in CD storage, increasing this way its use into the ministry and allowing to partially revert the high costs of the satellite images.

The Internet version is still in the testing phase, in process of catching all the user needs, requests, and problems of use, which will help us in the development of a next version of a web map server, now more ambitious, and based on the ideas proposed in the Web Map Server Interface Specification from the OpenGIS consortium.

References

[AAC99] G. Andrienko, N. Andrienko, J. Carter. "Thematic Mapping in the Internet: Exploring Census Data with Descartes". *TeleGeo'99, First International Workshop on Telegeoprocessing*. pp. 138--145. Lyon, France. May, 1999.

[BGS99] Tom Barclay, Jim Gray, Don Slutz. "Microsoft TerraServer: A Spatial Data Warehouse". 25th VLDB Conference 31-May-1999.

[BBBCS99] O.T. Balovnev, A. Bergmann, M. Breunig, A.B. Cremers, S. Shumilov. "Remote Access to Active Spatial Data Repositories". *TeleGeo'99, First International Workshop on Telegeoprocessing*. pp. 125--130. Lyon, France. May, 1999.

[CLH99]B. Cambray, C. Leclerc, J.R. Houllier. "Software Architectures Based on Cartographical Products". *TeleGeo'99, First International Workshop on Telegeoprocessing*. pp. 31--39 Lyon, France. May, 1999.

[ERMAP99] "ER Mapper Image Web Server". Earth Resource Mapping Pty Ltd. 1999

[ESRI98] Environmental System Research Institute, "*ESRI Shapefile Technical Description"* ESRI White Paper – July 1998

[FALBZM99] P. Fernández, P. Álvarez, M.A. Latre, R. Béjar, J. Zarazaga, Muro-Medrano "*Sistema de Información Geológico-Minero con capacidad de visualización SIG*" VIII Conferencia Nacional de Usuarios de ESRI 21-Oct-1999

[GC99] P.P. Gonzalves, M. Costa. "Local and Remote Geoprocessing Applications". *TeleGeo'99, First International Workshop on Telegeoprocessing*. pp. 53--60. Lyon, France. May, 1999.

[GNU97]*"A client/server JDBC Driver based on Java RMI"* http://dyade.inrialpes.fr/mediation/download/RmiJdbc/RmiJdbc.html

[HART97] R. Hartman, "*Focus on GIS Component Software. Featuring ESRI's MapObjects®*". OnWord Press. 1997.

[KP99] D. Kotzinos, P. Prastacos. "GAEA, a Java-based Map Applet". *TeleGeo'99, First International Workshop on Telegeoprocessing*. pp. 131--137. Lyon, France. May, 1999.

[LIMP99] W.F. Limp, "WEB MAPPING". *GEOEurope*. N. 8, pp. 18—22. Dec. 1999.

[OPENGIS99] Open GIS Consortium. "*OpenGIS Web Map Server Interface Specification*." 1999

[SM98] A. Sorokine, I. Merzliakova. "Interactive map applet for illustrative purposes". *Proceedings of the 6th International Symposium on Advances in Geographic Information Systems*. pp. 46—51. 1998.