A Generic Architecture for Geographic Information Systems*

Miguel R. Luaces, Nieves R. Brisaboa, José R. Paramá, and Jose R. Viqueira

Laboratorio de Bases de Datos
Facultade de Informatica, Universidade da Coruña
Campus de Elviña, s/n. 15071. A Coruña. Spain
+34 981 16 70 00 Ext. 1306
Fax: +34 981 16 71 60
{luaces, brisaboa, parama, joserios}@udc.es

Abstract. Geographic information systems (GIS) are becoming more usual due to the improved performance of computer systems. The traditional three-tier architecture for information systems is being used for the software architecture of GIS, but geographic information has some special qualities that make the use of the traditional architecture inappropriate.
We analyze in this paper the special characteristics of geographic information, and we take them into account to propose a generic architecture for geographic information systems. The result is an efficient system architecture that can be easily adapted to specific applications.
We also compare our architecture to the proposal of the OpenGIS Consortium, and we present a real-life geographic information system developed using commercial GIS development tools, and the architecture proposed in this paper. Finally, we analyze the impact that using commercial tools has on the system architecture.

1 Introduction

The exponential improvement in the performance of computer systems has made possible the design and development of tools to store, query and visualize geographic information, which can be as simple as the position in the map of all the hospitals in a country or as complex as the partition of the land of the country with regard to the kind of vegetation that grows on it. These tools are called *Geographic Information Systems (GIS)* [1].

After many years of research and development, it is now generally accepted that the architecture of information systems must consists of three separate layers, namely: the *presentation* layer, the *application logic* layer, and the *data* layer. This architecture allows for independence of the application from the data management subsystem and the data visualization techniques. For instance, if the relational data model is used as the topmost interface of the data layer in an application, it is possible to use any relational database management system as data source with little impact to the application.

There are some exclusive qualities of geographic information that make GIS different from traditional information systems. For instance, there is no support for geographic data types in the relational data model, geographic data sets tend to be large,

* This work was partially granted by CICYT (refs. TIC2002-04413-C04-04 and TIC2003-06593), and Xunta de Galicia (ref. PGIDIT02SIN10501PR).

and geographic information requires many different analysis and visualization procedures. These special qualities of geographic information must be taken into account when developing a GIS.

In order to achieve the same consensus as in the case of the traditional information systems architecture, the main goal of our work consists in defining a generic architecture for geographic information systems that adapts the traditional architecture of information systems to provide support for the special qualities of geographic information. Our strategy to achieve this goal consists in analyzing the special characteristics of GIS with respect to traditional information systems, and then designing a generic architecture for GIS that takes them into account. We describe these characteristics in Section 2, and the architecture is presented in Section 3.

There have been many research and industrial efforts to standardize many aspects of GIS technology, particularly by the OpenGIS Consortium (OGC) and the ISO. Their long-term goal consists in the full integration of geospatial data and geoprocessing resources into mainstream computing and the widespread use of interoperable geoprocessing software and geodata products throughout the information infrastructure [2]. The standards published by the OGC provide a precise and detailed description of many components of a GIS. In Section 4, we describe the OGC proposal for the architecture of a GIS, and we compare it to our proposal.

We have used the architecture proposed in this paper for the analysis, design and implementation of a GIS for a real-life problem, which is currently in use by the provincial council of A Coruña (Spain). The development of this GIS allows us to compare to what extent our architecture can be applied in a real-life situation using commercial tools. We describe in Section 5 the background of the project that initiated the development of the GIS and the system resulting from the development. Section 6 reports on the differences between the proposed architecture and the implemented system, analyzing the causes and the consequences of the differences. We end this paper giving some concluding remarks and describing future work in Section 7.

2 Special Characteristics of Geographic Information

In this section, we analyze the characteristics of geographic information which are peculiar to it, and we describe the requirements that these characteristics pose over the design of the architecture of geographic information systems.

The relational data model has been successful in defining a set of data types and operations that can be used to give support to a great extent of applications. This has made possible the creation of relational database management systems (relational DBMS) that provide physical and logical data independence to the applications using them. By using the relational data model in the architecture of an information system, it is possible to use the relational DBMS from different software vendors with little impact to the application.

However, the traditional types of the relational data model do not include data types for geographic data. If geographic information is represented using traditional

relational data types, the implementation of geographic operators and display procedures is complex and inefficient. Therefore, it is necessary to extend the definition of the relational model with new data types that represent geographic values, and new operations that work on these data types.

Once a data model supporting geographic data types and operations has been defined, it is important to find methods to display the values. Traditional relational data types can be displayed in a user interface using the same structures and abstraction levels that are used in the DBMS. For instance, integer numbers or string values can be displayed with text, and relations can be displayed as tables or record sets. However, geographic values represent complex structures that can only be displayed graphically in a map. The system architecture must implement the traditional metaphors for displaying relational data, and a newly-defined map metaphor to provide a graphical representation of the geographic information retrieved from the database. The user interface must be designed to allow the end-user to easily and efficiently access the geographic information.

Since a map metaphor is used in a system, it is necessary to distinguish between the objects in the DBMS and the objects that are displayed in the screen. Therefore, we need to introduce the following new terminology [3]. A *geographic object* is the representation in a DBMS of an object that occurs in the real world. It consists of a *spatial part* (i.e., the geometry describing its geographical extent) and a *descriptive part* (i.e., the alphanumeric attributes describing the object). On the other hand, a *cartographic object* is the graphical representation of a *geographic object* on a display.

Since the system must provide two different metaphors for displaying query results, it is natural that there are two different types of queries: *alphanumeric* and *geographic* queries. The result of an alphanumeric query is a relation with alphanumeric data to be displayed using a table or record-oriented metaphor, whereas the result of a geographic query is a relation with geographic data to be displayed using the map metaphor. The system must provide mechanisms to pose both types of queries to the DBMS using not only relational operators but also geographic predicates and functions. For instance, it must be possible to perform a *spatial join* in which the join condition is not an equality of two alphanumeric attributes, but rather a spatial predicate (e.g., checking whether two geographic attributes overlap). Moreover, since geographic objects cannot be given by the user as text, the user interface must provide tools to draw the spatial part of geographic objects to be used by the system as parameters in queries. Finally, since a geographic object can be displayed using two alternative metaphors, it is necessary that the system provides facilities to find the alternative view of a geographic object. It must be possible to retrieve the descriptive part associated to a spatial part selected by the user, or to locate in the map the spatial part associated to a descriptive part.

The map metaphor for displaying geographic information must support the different analysis and visualization procedures required by geographic information. First, it must offer the user tools to compose maps with the query results by adding or removing cartographic objects to the map, and changing the display styles and the order in which the objects are displayed. Then, it must be possible to change the display pa-

rameters of the map to focus on the appropriate phenomena, for instance by changing the map scale or the map limits.

In order to understand a map that results from a query posed by a user, it is necessary to provide some *context information* [4] to help the user to identify the results. Examples of context information are an *overview* map displaying the position of the map portion currently displayed, the *map scale* both as a numeric value and as a graphical scale bar, the *north arrow* describing the orientation of the map, or the position and extent of the displayed map. The system architecture must implement mechanisms to support these features.

Another common visualization procedure for geographic information are *thematic maps*, which consist in the representation of a set of geographic objects with different graphical styles according to the values of an attribute. For instance, a map that displays the countries of the world with different colors according to the population of the country.

The map metaphor requires that a geographic object is displayed using different cartographic objects in different situations [5,6]. The system architecture must provide support for this characteristic. Examples of uses of this feature are the following:

- A geographic object may be associated to multiple cartographic objects of different resolutions. When the map scale is small, it is not necessary that the cartographic object has the same resolution as the geographic object, because the extra detail cannot be displayed on the computer screen.
- A geographic object may be associate to multiple cartographic objects with different representations. For instance, a road is better represented as a surface at large scale maps, and as a line at small scale maps. A city can be represented as a set of surfaces representing the city blocks at a large scale map, as a single surface at a medium scale map, and as a point at a small scale map.

A final characteristic of geographic information that greatly influences the design of GIS is that geographic data sets tend to be large. The system architecture must be designed in such a way that the amount of data exchanged between the layers is kept as low as possible, while maintaining the quality of the map. This is more important when the user interface is not located at the same machine of the DBMS and the information must be transmitted over a network. Possible techniques include the following:

- It must be possible to select the map size and format used in the user interface. This reduces the size of the data set that is transmitted from the DBMS to the user interface.
- The system architecture must allow to implement some of the functionality of the user interface using cartographic objects instead of making DBMS requests. This reduces the amount data transmitted from the DBMS to the user interface.
- The multiple representation features described above must be used to transmit and display cartographic objects of the appropriate resolution. The system must also implement some mechanisms to manage the information density in the displayed map. This reduces the size of the data set that is transmitted.

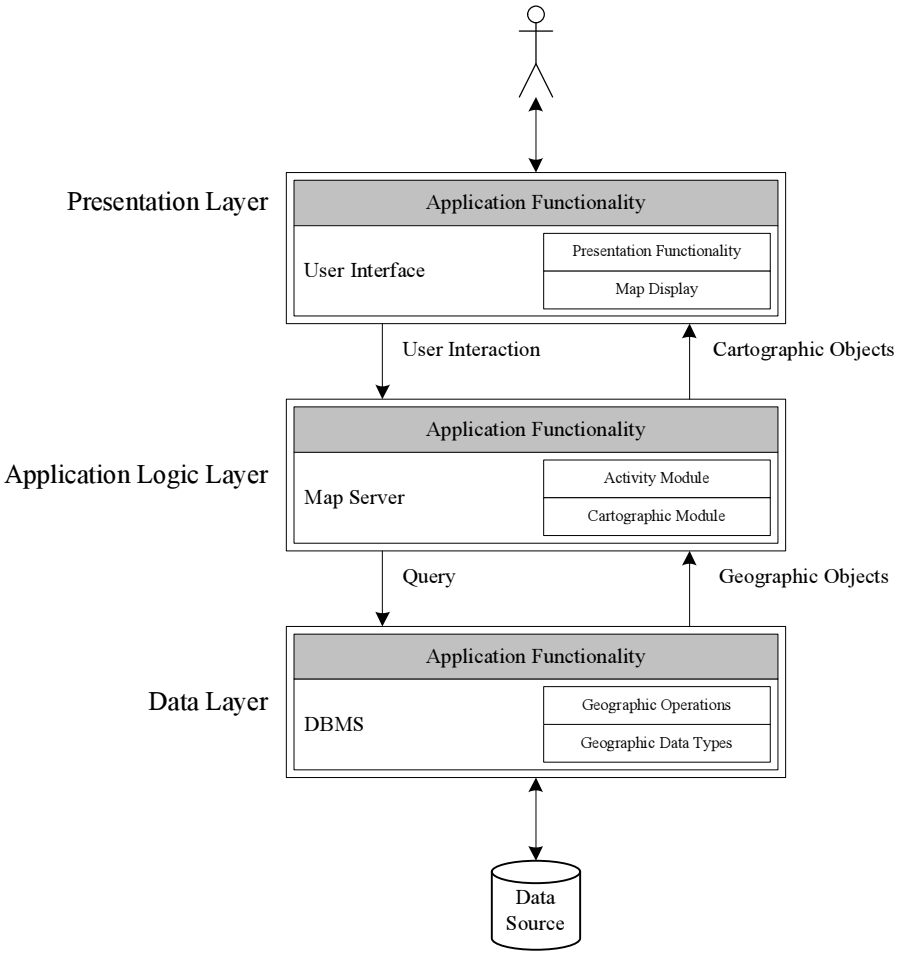

Fig. 1. A Generic System Architecture for GIS

3 Generic Architecture for a Geographic Information System

After analyzing the special characteristics of geographic information and the requirements that they pose over the architecture of geographic information systems, we propose in this chapter a generic architecture for GIS that meets these requirements. We describe the components of the architecture, and we illustrate its functionality through existing standards, commercial applications, or research results.

Figure 1 shows our proposal of a generic architecture for geographic information systems. It is a three-tier architecture comprising three main layers, namely the *Data* layer, the *Application Logic* layer and the *Presentation* layer. The Data layer provides data management functionality independently from the software technology. The Presentation layer is responsible for implementing the user interface of the system, displaying the maps and providing some functionality over them. Finally, the Application Logic layer implements the functionality of the system, using the Data layer to an-

swer the user requests and converting the geographic objects from the Data layer to cartographic objects to be displayed by the Presentation layer.

The functionality of these layers must be implemented independently of any particular application. The strategy consists in finding the features that are independent of the data schema and the application, and isolate them from the dependent ones. Then, these features are implemented using generic algorithms and a developer fills in the application-specific details. This is represented in the figure by the grayed modules in each layer containing the functionality specific to the particular application. We describe now each layer in more detail.

3.1 The Data Layer

The Data layer consists of a database management system that provides the data storage abstraction and the query language. The first commercial approaches to provide geographic data types and operators were implemented as a software layer on top of the DBMS. This has proven inefficient, and it is now accepted that the geographic data types and operators must be provided by the DBMS in order to achieve efficient data storage and algorithm implementations [7,8,9,10]. Therefore, the geographic data types and operators must be given as extension modules for the DBMS.

There has been many proposals in the literature for geographic data models. It is outside the scope of this paper to analyze here which one to choose (see [11] for an exhaustive survey and comparison). However, we may briefly describe some common aspects:

- There are two different and alternative views of space, the *field-oriented* and the *object-oriented* view of space [12]. Each view requires a different set of data types and operations. Both approaches are needed to completely describe geographic information.
- In the object-oriented view of space, data types are defined according to the dimensions of the real-life object that they represent. The data types are usually, *point*, *surface*, *surface* and *polyhedron*. In addition to these data types, geographic data models often define other data types to represent homogeneous and heterogeneous sets of objects. Data types to represent restricted cases of the general data types are usually defined as well.
- There is no generally accepted subset of primitive operations on these data types.

A particular application may need to provide some specific functionality in the Data layer. For instance, it is not possible that a DBMS extension module implements all operations that may ever be needed. Usually, only some basic operations are implemented, and the developer is expected to provide implementations for the advanced manipulation algorithms needed by a particular application.

The functionality of the Data layer is accessed by using the query language defined by the DBMS and the geographic extensions module. This query language can be a text-based high-level query language (e.g., SQL derivatives), or it can be a low-level query language based on an application programming interface. The result of a

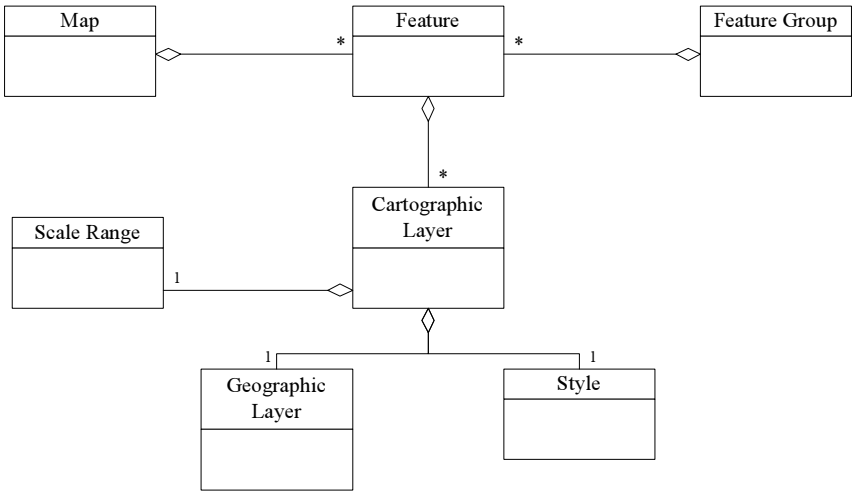

Fig. 2. Conceptual Model for the Cartographic Module

query is a set of data objects represented using the data abstraction of the DBMS. The result may be accessed again using a high-level, text-based descriptive language (e.g. GML [13]), or a low-level, application programming interface (e.g. OpenGIS Simple Features for CORBA [14]).

3.2 The Application Logic Layer

The Application Logic layer implements the functionality of the system. Particularly, it receives the requests of the user from the Presentation layer and translates them to the appropriate queries for the Data Layer. Then, it translates the geographic objects returned by the Data layer into the cartographic objects that are displayed by the Presentation layer. This task is performed by the *cartographic module*, which defines a mapping from geographic objects to cartographic objects using as parameters the map properties and the styles defined by the user. We define in this paper a cartographic module that fulfills the requirements presented in Section 2 by providing a new data abstraction that offers multiple cartographic objects for a geographic object in a transparent manner.

Figure 2 shows a conceptual model for the Cartographic Module. The classes *Feature*, *Feature Group* and *Map* are public and are a part of the interface of the Application Logic layer. The classes *Geographic Layer*, *Style*, *Cartographic Layer*, and *Scale Range* are private to the Cartographic Model.

We present now an example to help clarify these concepts. Imagine that we have a database that contains a relation with information for each road section (a geographic object). Each tuple of the relation consists of a set of alphanumeric attributes (descriptive part), and an attribute with the geometry of the surface of the road section (spatial part). In order to avoid computing, transmitting and displaying information that is not relevant, we want to display a road map that shows the road sections as a surface when

the map scale is large, as its center line when the scale is medium, and as a simplified version of the center line when the scale is small. At a small scale, the bends in the road are not significant anymore, and it makes no sense to transmit and display the unnecessary details.

The cartographic module must allow the developer to define a mapping from geographic objects to cartographic objects in such a way that the end-user manipulates only one object (a road in our example) while the representation in the map may vary in style, resolution and other aspects. To do this, a new data abstraction for cartographic objects, called *Feature*, is defined in the cartographic module. A *Feature* has, among other properties, a name. In our example, a developer defines a *Feature* named *roads*.

A *Feature* must have different graphical representations according to the map properties of the map that is being displayed (e.g., scale). Therefore, a *Feature* is associated to a set of *Cartographic Layer* objects. Each *Cartographic Layer* object encapsulates a set of cartographic objects and defines:

- the set of map properties that must hold for this *Cartographic Layer* to be visible in the map,
- the set of geographic objects that are displayed,
- and the style to be applied to the geographic objects to create the cartographic objects.

The properties that must hold for the objects to be visible are given using classes associated to *Cartographic Layer*. In Figure 2, we only show the class *Scale Range* that represents the definition of a range of map scales. If the map scale is within the range defined in the *Scale Range* associated to the *Cartographic Layer*, the cartographic objects are displayed in the map.

The geographic objects are fetched from the Data layer using a query that is represented by the class *Geographic Layer*. Each object consists of a query that retrieves the set of attributes that define each geographic object, and the name of the attribute in the query that defines the spatial part of the geographic objects. The remaining attributes in the query define the descriptive part.

Finally, the style that is applied to the geographic objects is defined using the class *Style*. There is a specific subclass of *Style* for each geographic data type (which are not shown in the figure), and each class consists of a set of properties defining the graphical representation of each geographic object (e.g., a line color, a line style, a fill color, or a fill pattern).

In our example, three *Cartographic Layer* objects are needed, one for roads in a large scale map that displays the surface of the road section, another for roads in a medium scale map that displays the center line of the road section, and a third one for roads in a small scale map that displays a simplified version of the center line. Each *Cartographic Layer* object is associated to a *Scale Range* object, a *Geographic Layer* object, and a *Style* object.

For large scale roads, we define a *Geographic Layer* with a query that fetches the tuples of the relation. The spatial part of the geographic objects is defined by the ge-

ographic attribute, and the descriptive part consists of the remaining attributes. The *Geographic Layer* for medium scale roads consists of a query that fetches the alphanumeric attributes of the relation and derives a new geographic attribute applying a function to compute the center line of the geometry. For small scale roads, the *Geographic Layer* is defined using a similar query to the previous one, but deriving the geographic attribute using an additional function to simplify geometries.

By defining and composing the appropriate *Cartographic Layer* objects for different combinations of map properties, a developer can build *Feature* objects that have different cartographic representations for different map properties. The Application Logic layer and the Presentation layer interact with *Feature* objects, and therefore the system handles multiple cartographic representation transparently. The end-user interacts with *Feature* objects as well, and they are used to load, display, hide and remove geographic data from the map.

The last two classes of the model in Figure 2 are utility classes for the Application Logic layer and the Presentation layer. The class *Map* is used to predefine maps as sets of *Feature* objects that are useful to the application. Instead of making the end-user or the application code to build a map from scratch, the map definitions can be created and stored in advance. In our example, a developer defines a *Map* object consisting of the *Feature* roads and any other features that are of interest for the application. When the end-user chooses one of the predefined maps, its features are loaded automatically and the user does not have to build the map from scratch.

3.3 The Presentation Layer

The maps computed by the Application Logic layer are displayed by the Presentation layer. This layer also displays the graphical controls that receive the interaction from the user, and converts these interactions to requests for the Application Logic layer.

The task of displaying the map is carried out by a *map display* component, which receives a set of cartographic objects and displays them in the screen. The component also displays buttons and other controls to manipulate the graphical representation of the map.

If all the functionality of the system is implemented in the Application Logic layer the response time of the system may not be fast enough. This is because each user action may require a set of requests to the underlying layers, which must then be answered, and the result sent back to the Presentation layer. In order to achieve faster response time from the system, the Presentation layer may implement some functionality using the cartographic objects for the computations. This is implemented in the *presentation functionality* module, and consists in tools for the following tasks:

- Control of the map scale and position. This allows the end-user to focus on the map phenomena of interest by zooming and moving the map.
- Management of the graphical legend. It allows to understand the map by describing the real-world entities that are described by each symbol. It also allows to the end-user to customize the map by adding and removing additional features.

– Measure distances and areas.
– Display the map context using an overview map, the map scale using a graphical scale bar, the map orientation using a north arrow, the map center using the longitude and latitude values, and the map extent using the width and height values.
  Another important tool to display the map context consists in a dynamic label for the cartographic objects that appears when the mouse is moved over the object.

Finally, developers must implement the advanced interaction mechanisms that are specific to the particular application that is being developed. This is done in the *application functionality* module.

4 The OpenGIS Consortium

The OpenGIS Consortium (OGC) has devoted many efforts to define standards for many aspects of GIS technology. The result is a large set of abstract and implementation specifications that are used by many commercial and research GIS development tools. The goal of our work is to some extent similar to the goal of the OGC. Therefore, in this chapter we compare our work to the standards of the OGC, and we analyze their similarities and differences, and the improvements that our proposal enables.

The OGC produces two types of specification products: *abstract specifications* and *implementation specifications*. The former defines a conceptual model to be used as a basis for the creation of the latter, which consists in an unambiguous specification ready for the implementation.

The basis of the architecture defined by the OGC is the definition of a geographic data model [15] consisting of geographic data types and operators. This abstract specification has been used to create many implementation specifications for different technologies [14,16,17]. These specifications are used in many commercial applications as the geographic data model, and it can be used for our architecture proposal as well.

Using this specification as a basis the OGC defines a general-purpose data model that consists of *features, feature types* and *feature collections* [18]. A feature is an abstraction of a real world phenomenon. A geographic feature is a feature associated with a location relative to the Earth. A feature type defines a class of features. Finally, a feature collection is a special category of feature that represents a group of features that have common metadata and formal relationships.

The architecture of a GIS according to the OGC consists of a set of services that interact at interfaces. The OGC defines an comprehensive set of services for multiple purposes, for instance:

– The Web Mapping Services (WMS) [19] standardizes the way in which clients request maps. Clients request maps from a WMS instance in terms of named layers and provide parameters such as the size of the returned map as well as the spatial reference system to be used in drawing the map.

– The Web Feature Service (WFS) [20] supports the manipulation of geographic features, including insertion, update, deletion and querying. The output of the WFS consists in GML representations [13] of the query results.
– The Web Coverage Service (WCS) [21] supports the interchange of field-oriented data.
– The Coverage Portrayal Service (CPS) [22] defines a standard interface for producing visual pictures from coverage data. This service extends the WMS interface and uses the Styled Layer Descriptor (SLD) language to support rendering of WCS coverages.

The approach followed by the OGC to build has its major advantage in the orientation to a service-based architecture for GIS. This orientation allows for highly-distributed architecture because services only interact at the interfaces, and therefore they can be located anywhere in a network. Our proposal is highly influenced by the OGC approach, mainly on the organization of the tasks and the strict separation of the modules that interact only using well-defined interfaces.

The main drawback of the OGC proposal is that it is not finished, and there are many pieces missing in the architecture. Moreover, the definition of specifications by multiple groups working independently may cause that the specifications do not match perfectly due to coordination problems.

5 The EIEL Geographic Information System

In order to prove the applicability of our proposal, we have developed a GIS using the system architecture and the concepts that we describe in Section 3. We have decided to use a real-life problem to ensure that we face actual problems that can be found in the development of GIS. We have also decided to use commercial tools instead of custom-made tools as much as possible to avoid the increase in costs and time that custom-made tools imply.

We give first some background information regarding the project of which the GIS is a part, then we present some details of the system architecture of the GIS.

5.1 The EIEL Project

To optimize the management of the country, the Spanish territory is divided in four levels, each having its own government structures with its own areas of responsibility. The central government deals with issues concerning the entire country, such as foreign affairs. The country is then divided into nineteen autonomous regions (*Comunidad Autónoma*), each having its own government with different levels of independence. The government of an autonomous region has legislative, executive and judicial power over the areas whose competence was delegated by the central government (e.g., health care or education). Each autonomous region is further divided into a variable number of provinces (ranging from one province in the smallest ones to nine in the biggest one).

Each province has its own provincial council (*Diputación Provincial*) whose tasks are mainly of an administrative nature. Finally, each province is divided into a variable number of municipalities.

The provincial council is in charge of coordinating the work of the municipalities in the province, ensuring solidarity and balance among them. It encourages the economic development and improves living conditions in underdeveloped municipalities in order to avoid big differences between them. These goals are accomplished by realizing the following tasks:

- Ensure the correct provision of municipal services in the entire province (e.g.: social services, cultural services).
- Coordinate the services of the different municipalities among them.
- Give legal, economic and technical assistance and cooperation to municipalities, especially to those with less economic and management capacity.
- Render the public services that imply several municipalities, (e.g., water supply, sewage treatment, road maintenance).
- Take part in the coordination process between the municipal government and the regional and country governments.

In order to discover the funding needs of each municipality and to propose special action programs to balance the living conditions of the municipalities, the provincial councils need a tool to objectively analyze and evaluate the situation and state of infrastructure and equipment in each municipality. For this purpose, the national government requires every provincial council to conduct, every five years, a survey on local infrastructure and equipment, (named EIEL from the Spanish *Encuesta de Infraestructura y Equipamientos Locales*). The national government defines a common methodology to be followed by all provincial councils that results in a database with standardized and homogeneous information in terms of format, time, geographic scope, and evaluation criteria. This information can be used to compute indicators and define mechanisms to evaluate and compare the state of the infrastructure and equipment in the municipalities. This allows to discover deficiencies in services and act accordingly.

The provincial council of A Coruña decided to broaden the goals of the EIEL for the year 2000. The province of A Coruña is located in northwestern Spain. With one million inhabitants in excess and almost eight thousand square kilometers, it is densely populated with more than a hundred and twenty-five inhabitants per square kilometer. Particularly, these new goals were considered:

- Extend the information to be collected, both in terms of the different kinds of elements to be surveyed, and the amount of information for each particular item.
- Reference the items surveyed to its geographical location or extent.
- Build an information system with the information collected to be used by the provincial council staff, and build a publicly-accessible, web-based information system.

This was achieved through a two-year project carried out by the University of A Coruña. A large group of students from the civil engineering school and the architecture school, supervised by a group of professors, collected the data by direct observation or interviewing the responsible staff in the municipality. At the database group of the University of A Coruña, we designed and developed the applications supporting the data collection work flow. Then, we developed a geographic information system to manage and exploit these information.

The result of this project is a powerful analysis tool in which many different evaluation mechanisms can be designed and many indicators can be computed to analyze the living conditions in the province. The existence of geographic data attached to the descriptive data allows to perform geographical analysis in the information. Moreover, the existence of a database with the information for the year 2000 and a set of tools for the management of this information, turns the survey of the year 2005 into a much easier work.

A detailed description of the database schema for the geographic information system is outside the scope of this paper. We refer the interested reader to [23]. Instead, we enumerate here briefly the types of information collected, and we present some statistical information about the database.

- Territorial structure, collaboration relationships and urban planning. We have collected information about 4000 towns and villages.
- Road network. More than 35000 road sections and more than 40000 street sections were surveyed.
- Water supply and sewage treatment (approximately 17000 pipe sections each).
- Waste disposal.
- Facilities in the municipality, including among others 700 education facilities, 1200 sport facilities, 1000 parks and gardens, 800 culture facilities, and 200 health centers.

5.2 System Architecture

For the implementation of the GIS briefly described in the previous section, we have tried to apply the architecture proposed in Section 3. However, since we wanted the system to be ready in as little time as possible, we have also tried to use as much functionality already existing as possible. This implies that existing applications were preferred over new development projects.

Figure 3 shows the system architecture of the application. The components that we had to develop are grayed, whereas the commercial components are shown with a white background. The system is web-based, so that it can be used both by the staff at the provincial council through an intranet, and by the citizens on the Internet.

On the server side, the data is managed by a relational database management system (Microsoft SQL Server 7.0) and the geographic data model is provided by Intergraph Geomedia Web Map. The cartography and the activity modules had to be

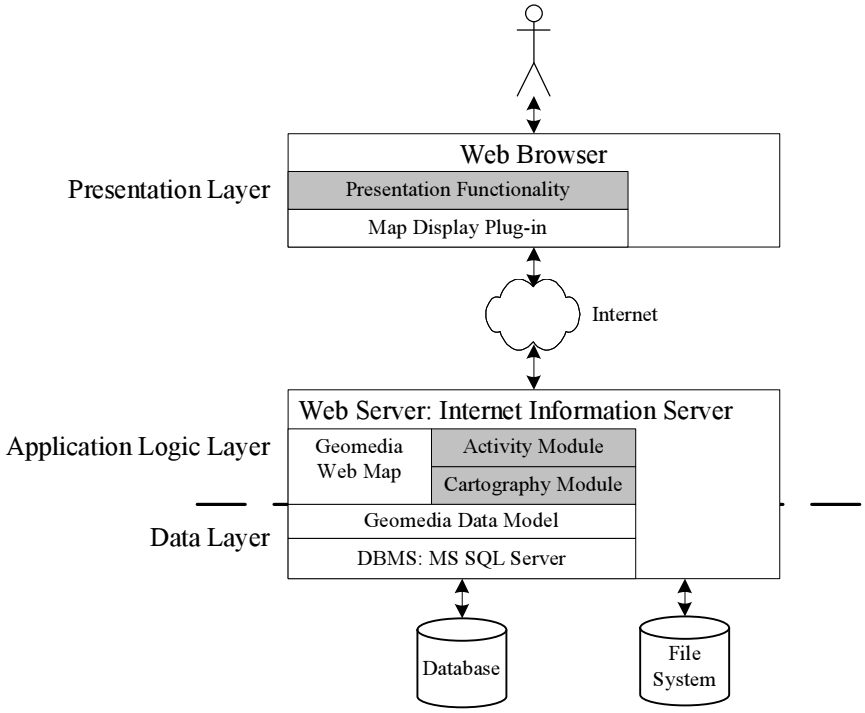

Fig. 3. System Architecture for the EIEL GIS

developed for this application because the ones provided by Geomedia were not sufficient to meet our requirements. The information of the web application is served by Microsoft Internet Information Server, the web server supplied by Microsoft. This particular server was chosen because it is required by Geomedia Web Map.

On the client side, any web browser can be used that supports the HTML dialect of our application. The map is displayed using a Java plugin provided by Intergraph, which also implements part of the Presentation functionality. The remaining functionality is implemented by custom-developed Javascript modules.

6 Analysis of the EIEL System Architecture

The use of commercially available tools in the development of the GIS described in the previous section caused that the architecture of the resulting system had to differ at many points from the architecture we propose. We compare both architectures in this section, and we analyze the causes of the differences, and their consequences.

The main difference between the architecture we propose in Section 3 and the system we implemented is that the geographic data model provided by Geomedia Web Map is not implemented within DBMS. Instead, it is implemented as a layer on top of the DBMS. Geographic information is stored using large objects of the DBMS, and geographic operations are implemented in main memory using the results returned by

the DBMS. This causes the geographic data model provided by Geomedia Web map to be terribly inefficient.

As an example, the computation of a geographic join (i.e., a join between two relations using a geographic predicate) requires both relations to be loaded completely into main memory, its Cartesian product to be computed, and a selection using the geographic predicate to be performed. This is much more inefficient than a join performed at the DBMS using indexes.

Moreover, Geomedia Web Map imposes a proprietary data storage format which breaks our data independency requirement. If a standard like [16] were used for the storage of geographic information, we could switch the GIS development tool with little impact to the application. In addition to that, Geomedia Web Map forces us to use a particular web server. The result is that our application is too heavily integrated with the technology and cannot be ported to other software platforms.

7 Conclusions and Future Work

The contribution of this paper is a generic architecture for geographic information systems. We have based the design of the architecture on the analysis of the special characteristics that make geographic information system different from traditional information systems. Then, we have compared briefly our architecture to the proposal of the OpenGIS Consortium, which can be considered the yardstick for GIS. Finally, we have described a real-life project for which we have applied the architecture we propose. We have tried to use commercial application in the development process of this project, and we have analyzed in this paper the consequences that these tools have in the system architecture.

The next steps in the line of work suggested by this paper are:

- Implement the architecture. Even though we have already implemented parts of the architecture for the project described in Section 5, this implementation is not generic because it uses Intergraph Geomedia Web Map. We plan on implementing the architecture using standards for the interfaces of the system
- Develop tools to create geographic information systems based on the architecture. Once the architecture is completely implemented, it is possible to build specific applications by giving its details using high-level languages or using visual interfaces.

References

1. Laurini, R., Thompson, D.: Fundamentals of spatial informations systems. The APIC Series Nř 37 - Academic Press (1992) ISBN: 0-12-438380-7.
2. Open GIS Consortium: OpenGIS Abstract Specification. Topic 0: Abstract Specification Overview. OpenGIS Project Document 99-100r1, Open GIS Consortium, Inc. (1999)
3. Voisard, A.: Designing and integrating user interfaces of geographic database applications. In: Proceedings of the ACM Workshop on Advanced Visual Interfaces. (1994) 133–142

4. Egenhofer, M.J.: Interaction with geographic information systems via spatial queries. Journal of Visual Languages and Computing **1** (1990) 389–413
5. Rigaux, P., Scholl, M.: Multiple representation modelling and querying. In: IGIS. (1994) 59–69
6. Frank, A., Timpf, S.: Multiple representations for cartographic objects in a multi-scale tree - an intelligent graphical zoom (1994)
7. Davis, J.: IBM's DB3 Spatial Extender: Managing Geo-Spatial Information within the DBMS. Technical report, IBM (1998)
8. Adler, D.W.: IBM DB2 spatial extender - spatial data within the RDBMS. In: Proceedings of the 27th International Conference on Very Large Data Bases(VLDB '01), Orlando, Morgan Kaufmann (2001) 687–692
9. Brisaboa, N.R., Lema, J.A.C., Luaces, M.R., Viqueira, J.R.: State of the art and requirements in gis. In: Proc. of the Third Mexican International Conference on Computer Science (ENC'01), Aguascalientes, Mexico (2001)
10. Koubarakis, M., Sellis, T.K., Frank, A.U., Grumbach, S., Güting, R.H., Jensen, C.S., Lorentzos, N.A., Manolopoulos, Y., Nardelli, E., Pernici, B., Schek, H.J., Scholl, M., Theodoulidis, B., Tryfona, N., eds.: Spatio-Temporal Databases: The CHOROCHRONOS Approach. Volume 2520 of Lecture Notes in Computer Science. Springer (2003)
11. Viqueira, J.R.: Formal Extension of the Relational Model for the Management of Spatial and Spatio-temporal Data. PhD thesis, Universidade da Coruña, Spain (2003)
12. Couclelis, H.: People manipulate objects (but cultivate fields): Beyond the raster-vector debate in gis. In Frank, A.U., Campari, I., Formentini, U., eds.: Theories and Methods of Spatio-Temporal Reasoning in Geographic Space: Proc. of the International Conference GIS. Springer, Berlin, Heidelberg (1992) 65–77
13. Open GIS Consortium: OpenGIS Geography Markup Language (GML) 3.0 Implementation Specification. OpenGIS Project Document 02-023r4, Open GIS Consortium, Inc. (2003)
14. Open GIS Consortium: OpenGIS Simple Features Specification For CORBA. Revision 1.0. OpenGIS Project Document 99-054, Open GIS Consortium, Inc. (1998)
15. Open GIS Consortium: OpenGIS Abstract Specification. Topic 1: Feature Geometry (ISO 19107 Spatial Schema). OpenGIS Project Document 01-101, Open GIS Consortium, Inc. (2001)
16. Open GIS Consortium: OpenGIS Simple Features Specification For SQL. Revision 1.1. OpenGIS Project Document 99-049, Open GIS Consortium, Inc. (1999)
17. Open GIS Consortium: OpenGIS Simple Features Specification For OLE/COM. Revision 1.1. OpenGIS Project Document 99-050, Open GIS Consortium, Inc. (1999)
18. Open GIS Consortium: OpenGIS Abstract Specification. Topic 5: Features. OpenGIS Project Document 99-105r2, Open GIS Consortium, Inc. (1999)
19. Open GIS Consortium: OpenGIS Web Map Service Implementation Specification. OpenGIS Project Document 01-068r3, Open GIS Consortium, Inc. (2002)
20. Open GIS Consortium: OpenGIS Web Feature Service Implementation Specification. OpenGIS Project Document 02-058, Open GIS Consortium, Inc. (2002)
21. Open GIS Consortium: OpenGIS Web Coverage Service Implementation Specification. OpenGIS Project Document 03-065r6, Open GIS Consortium, Inc. (2003)
22. Open GIS Consortium: OpenGIS Coverage Portrayal Service Implementation Specification. OpenGIS Project Document 02-019r1, Open GIS Consortium, Inc. (2002)
23. Brisaboa, N.R., Lema, J.A.C., Martínez, A.F., Luaces, M.R., Viqueira, J.R.: The e.i.e.l. project: An experience of gis development. In: Proc. of the 9th EC-GI & GIS Workshop (ECGIS '03), A Coruña, Spain (2003)