411

UNIVERSIDADE DA CORUÑA
DEPARTAMENTO DE COMPUTACIÓN

A GENERIC ARCHITECTURE FOR GEOGRAPHIC INFORMATION SYSTEMS

Tese Doutoral
A Coruña, July 2004

Doutorando:	Miguel Ángel Rodríguez Luaces
Director:	Dr. Nieves Rodríguez Brisaboa
	Dr. José Ramón Paramá Gabía

UNIVERSIDADE DA CORUÑA
DEPARTAMENTO DE COMPUTACIÓN

A GENERIC ARCHITECTURE FOR GEOGRAPHIC INFORMATION SYSTEMS

Tese Doutoral
A Coruña, July 2004

Doutorando:	Miguel Ángel Rodríguez Luaces
Director:	Dr. Nieves Rodríguez Brisaboa
	Dr. José Ramón Paramá Gabía

Abstract

Geographic information systems (GIS) have been an active research field for decades now, and they are becoming a very active commercial field due to the increasing demand for interactive manipulation and analysis of geographic information and the exponential improvement in the performance of computer-based technologies. On the other hand, after many years of research and development, it is now generally accepted that the functionality of a generic architecture for information systems must be divided into three separate tiers, namely: *presentation tier*, *application logic tier*, and *data tier*. Even though this architecture is suitable for GIS applications, the special nature and exclusive characteristics of geographic information impose special requirements on the architecture in terms of conceptual and logical models, data structures, access methods, analysis techniques, and visualization procedures.

The main goal of our work consists in proposing a generic architecture for GIS that provides support for the special nature of geographic information and the functional requirements of GIS applications. Our strategy to achieve this goal consists of two steps: (i) we analyze the special characteristics of GIS with respect to traditional information systems, (ii) and we adapt the traditional three-tier architecture for information systems to take into account the special characteristics of GIS.

Considering the goal of our work, we must review thoroughly the proposal of the OpenGIS Consortium and the ISO Technical Committee 211 pointing out their strengths and their drawbacks. We also compare the architecture and the solutions that we propose to the standards and specifications defined by these organizations. Finally, we applied the architecture that we propose in the development of a complete and complex real-life GIS using commercial tools in the analysis, design and implementation. We use this application to analyze the limitations of current commercial GIS development tools that cause differences in the architecture of the resulting system with respect to our proposal.

Resumen

Los sistemas de información geográfica (SIG) han sido un área de investigación muy activa durante décadas, y su aplicación comercial está ganando importancia debido al interés creciente por el análisis y manipulación de información geográfica y el incremento exponencial de la potencia de las sistemas de computación. Por otra parte, tras muchos años de investigación y desarrollo, se acepta que la funcionalidad de una arquitectura genérica para sistemas de información debe estar dividida en tres capas: *capa de presentación*, *capa de lógica de aplicación*, y *capa de datos*. Aún cuando esta arquitectura se puede aplicar a aplicaciones de SIG, la naturaleza especial y las características exclusivas de la información geográfica imponen requisitos especiales en la arquitectura con respecto a modelos conceptuales y lógicos, estructuras de datos, métodos de acceso, técnicas de análisis, y procedimientos de visualización.

Nuestro objetivo principal es proponer una arquitectura genérica para SIG que tenga en cuenta la naturaleza especial de la información geográfica y los requisitos funcionales de las aplicaciones de SIG. Nuestra estrategia para conseguirlo consiste en: (i) analizar las características especiales de los SIG con respecto a los sistemas de información tradicionales, (ii) y adaptar la arquitectura tradicional de sistemas de información en tres capas para tener en cuenta las características especiales de los SIG.

Dado el objetivo de nuestro trabajo, debemos revisar exhaustivamente la propuesta del OpenGIS Consortium y el ISO Technical Committee 211 mostrando sus virtudes y sus defectos. También comparamos la arquitectura y las soluciones que proponemos con los estándares y las especificaciones que proponen estas organizaciones. Finalmente, hemos aplicado la arquitectura que proponemos en el desarrollo de un SIG completo y complejo para un problema real, utilizando herramientas comerciales en el análisis, desarrollo e implementación. Utilizamos esta aplicación para analizar las limitaciones de herramientas actuales para el desarrollo de SIG que provocan diferencias en la arquitectura del sistema resultante con respecto a nuestra propuesta.

Acknowledgements

I am very grateful to Nieves R. Brisaboa for letting me work six years under her supervision at the Database Laboratory of the University of A Coruña, and above all, for trusting me all these years. I am also grateful to Jose Ramón Paramá for his constructive criticism that has raised the quality of this work. I will never forget my colleagues at the Database Laboratory of the University of A Coruña: J.R., Fari, Fran, Eva, Ángeles, Tony, Jose, Miguel, Raquel, Cris, y mon. They have made the idea of going every day to work appealing. I am also very grateful to the people that worked on the EIEL project, particularly to Carmen García and to Alberto Varela, because together we have achieved something great.

I am also very grateful to Prof. Dr. Ralf Hartmut Güting for giving me the first opportunity to start this work, and for his help and support during the two-year period that I worked under his supervision in Germany. I will never forget my colleagues and friends in Germany and all the things they did for me. Particularly, Stefan Dieker and our productive discussions regarding work and life in general, Anne Jahn, because without her my life in Germany would have been much more complex, and Thomas Meilwes, who taught me to appreciate the German culture and language, and at the same time, the Spanish ones.

But above all, I want to thank my parents, Andrés and Mari, because without their support, patience, and love, I would never have reached this goal. I will never forget either the confidence and the support of Paco and Maruchi, who have taken care of me as if I was another son.

Finally, if this work has been possible is because of Ana. Without her, I would have given up a long time ago. Thank you for all the times that you have encouraged me to continue when I wanted to give up. Thank you for your patience all those nights that you went to bed without me, and you woke up without me. In fact, there are no words to describe everything you have done for me. Thank you for everything. I love you.

Agradecimientos

Gracias a Nieves R. Brisaboa por permitirme trabajar bajo su dirección estos seis años en el Laboratorio de Bases de Datos, y sobre todo, por la confianza depositada en mí. También le estoy agradecido a José Ramón Paramá por su crítica constructiva que ha elevado la calidad de este trabajo. Nunca olvidaré a mis compañeros del Laboratorio de Bases de Datos de la Universidade da Coruña: J.R., Fari, Fran, Eva, Ángeles, Tony, Jose, Miguel, Raquel, Cris, y mon. Ellos han conseguido que me apetezca ir a trabajar cada día a Coruña. También estoy agradecido a toda la gente que trabajó en la EIEL, sobre todo a Carmen García y a Alberto Varela, porque juntos hemos conseguido algo estupendo.

También quiero agradecer a Ralf Hartmut Güting haberme dado la primera oportunidad para comenzar este trabajo, y la ayuda y el apoyo que recibí durante los dos años que estuve trabajando junto a él en Alemania. Nunca olvidaré mis compañeros y amigos de Alemania, ni todo lo que hicieron por mí. En particular, a Stefan Dieker y nuestras productivas discusiones acerca del trabajo y la vida en general, a Anne Jahn, ya que sin ella mi vida en Alemania hubiera sido infinítamente más difícil, y a Thomas Meilwes por enseñarme a apreciar la cultura y la lengua alemana, y con ello, también la cultura y la lengua española.

Pero sobre todo, quiero dar las gracias a mis padres, Andrés y Mari, porque sin su apoyo, paciencia y cariño no habría llegado hasta aquí. Nunca olvidaré tampoco la confianza y la ayuda de Paco y Maruchi, que me han cuidado como a un hijo más.

Finalmente, todo este trabajo ha sido posible gracias a Ana. Sin ella me habría rendido hace mucho tiempo. Gracias por todas las veces que me has empujado para que siguiera cuando me quería rendir. Gracias por tu paciencia todas esas noches que te has acostado sin mí, y todas esas noches que te has levantado sin mí. Realmente, no hay palabras para describir todo lo que me has ayudado. Gracias por todo. Te quiero.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

Geographic information, in the form of paper maps, has been a driving force behind the progress of our society for many centuries. Until a few decades ago, manipulating, synthesizing and representing geographic information was restricted to paper maps and these tasks were limited to manual, non-interactive processes. The exponential improvement in the performance of computer-based technologies and the increasing demand for interactive manipulation and analysis of geographic information have created a need for *geographic information systems* (GIS).

Many definitions for the term GIS have been proposed in the literature, each considering the functionality of the system from different perspectives. All of them focus on three different aspects:

1. a GIS is a *collection of computer tools to perform geographic analysis and simulations* [LT92, BM98, RSV01],

2. a GIS is supported by a set of *data structures and algorithms to represent, retrieve and manipulate geographic information* [Wor95, RSV01],

3. and a GIS is a utility that *helps people make decisions in tasks related to geography* [LGMR01, HA03].

We can extract our own definition from these ones by saying that a GIS is a computer-based system to efficiently *model, capture, store, manipulate, query, retrieve,*

This work was partially supported by the CHOROCHRONOS project, funded by the EU under the Training and Mobility of Researchers Programme, Contract No. ERB FMRX-CT96-0056, and the Xunta de Galicia (ref. PGIDIT02SIN10501PR).

analyze and visualize information, where part of the information is of a geographic nature.

An important characteristic of geographic information systems is that they are more than tools to produce paper maps. Whereas in traditional cartography the paper map is the database, in a GIS the map is only a projection of a particular view of a geographic database at a given time. This enables the GIS end-user to review an unlimited number of analysis alternatives, and to make maps from different points of view emphasizing different aspects of the information [BM98]. As a consequence, functional requirements for GIS are vast and go far beyond those of traditional information systems. For instance, public administration and market analysis applications require integrating and displaying data at different levels of resolution, fleet management applications need path-finding operations in transportation networks, and environmental information systems dealing with terrain analysis need data interpolation.

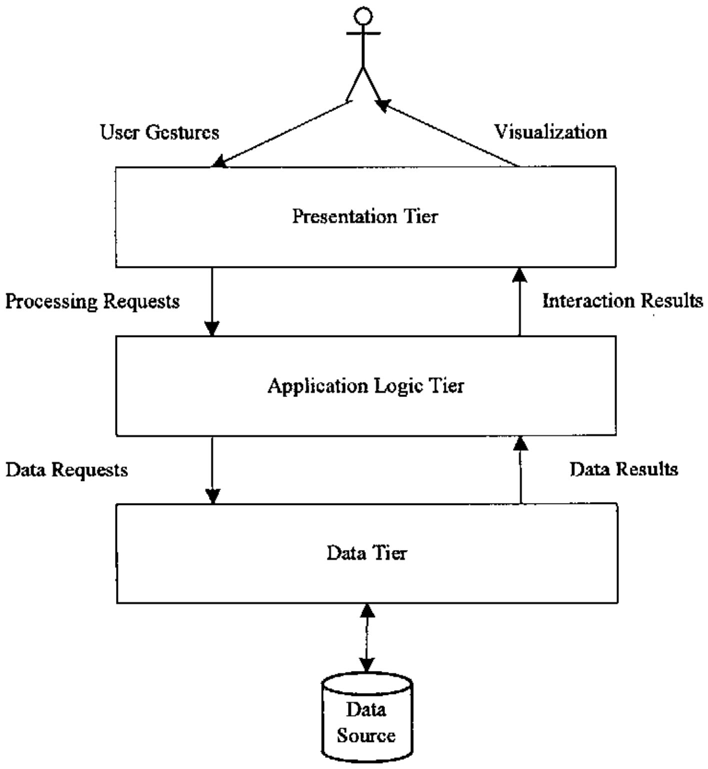


Figure 1.1: Three-tier Information System Architecture.

After many years of research and development, it is now generally accepted that the architecture of general-purpose information systems must consist of three separate tiers, namely: the *presentation tier*, the *application logic tier* (or *business logic tier*), and the *data tier* (see Figure 1.1). The main advantage of this architecture is that it enforces a strict separation of the functionality of the system into three different modules that interact only at interfaces. This enables a developer to modify each one of these modules of the application with little impact on the others. Therefore, this architecture provides increased performance, flexibility, maintainability, reusability and scalability. For instance, if the relational data model is used in the data tier of a general-purpose information system, it is possible to use any relational database management system (DBMS) as data source with little impact to the application.

Even though the three-tier architecture for general-purpose information systems is suitable for GIS, the special nature and exclusive characteristics of geographic information impose special functional requirements on the architecture of GIS applications in terms of conceptual and logical models, data structures, access methods, analysis techniques, and visualization procedures. For instance:

- *Special data types and operations are needed to represent and manipulate geographic information.* The data types available in logical models for traditional information systems are suitable for representing business information such as numbers, texts, or dates. However, the representation of geographic information on a computer requires new data types such as *point*, *curve* or *surface*. Similarly, the operations provided by logical models for traditional information systems manipulate values of business data types (e.g., arithmetic operations, or basic string processing). In order to manipulate geographic information, new operations are required (e.g., *distance*, *direction*, or *intersection*).

- *Geographic information requires many different analysis and visualization procedures.* There is a rich set of analysis techniques applicable to geographic information that require special transformation and analysis procedures including *path-finding algorithms in topological spaces*, *terrain analysis*, or *geographic data interpolation*. Moreover, given that geographic information requires specific data types and operations for its representation and manipulation, new visualization techniques are also required to display values of these new data types and results from these operations.

  In addition to this, geographic information has some peculiarities that have an impact in the presentation process, such as the need for different abstractions for representation and visualization. In a traditional relational database, a similar abstraction is used for the logical model (a relation) and the visualization of values (a table). On the other hand, the abstraction used in the logical model of a GIS for a data element (e.g., a point that represents a city) is different from the abstraction used in the visualization (e.g., a symbol, a point, or a small surface).

- *Geographic information is typically voluminous with a naturally imposed hierarchical structure.* Whereas the representation of traditional business data requires few storage bytes, the representation of geographic data requires larger amounts of storage space. For instance, to represent the border of a country, thousands of pairs of real numbers are needed. This implies that efficient storage representations, access structures, and operation algorithms must be defined for geographic information. Furthermore, the special nature of geographic space imposes relationships and a hierarchy between spatial values that do not exist in other data domains. For instance, roads are related by a connection relationship and buildings in a city are related by a containment relationship to the city block in which they are. These relationships are important and must be represented in the system.

- *Geographic information processing is characterized by transactions that are much longer than a typical standard relational database transaction.* This follows from the previous characteristic. Given that geographic information is voluminous, the creation and modification of geographic data requires much more time than the same tasks for traditional business data. Therefore, long transactions are required to keep the data locked for a longer period of time.

- *There are two different conceptual views of geographic space.* It is widely recognized that geographic space may be conceptualized in two distinct ways: using *field-based* models or using *object-based* models. Field-based models represent geographic information as collections of spatial distributions where each distribution may be formalized as a mathematical function from a spatial framework to an attribute domain. In these models, the geographic space is considered as an element existent in itself that has properties associated to each location. On the other hand, object-based models represent geographic information as discrete and identifiable entities that consist of a collection of attributes that describe the properties of a real-world phenomenon using alphanumeric and geometric values. These models consider the geographic space as a container for objects that have properties associated. Each view is conceptually different and needs to be represented in a conceptual model with different abstractions.

- *Each conceptual view of space can be represented in many different ways in a computer.* First, the logical models for traditional information systems do not provide efficient support for the representation of geographic information. Therefore, new logical models must be defined, or the existing ones must be extended in order to support geographic information. And second, each of the conceptual models of geographic information (i.e., the field-based model and the object-based model) can be represented using a logical model in many different ways.

These and other features impact the overall architecture of a GIS. Therefore, it is very important to determine the special requirements of GIS and the functionality that must be provided by GIS beyond the standard requirements and functionality of a general-purpose information system. This will enable us to design and implement a GIS with appropriate capabilities for modeling, collecting, querying, and visualizing geographic information.

1.2 Scope of the Thesis

The main goal of our work is a proposal of a generic architecture for geographic information systems that provides support for the special characteristics of geographic information. Our strategy to achieve this goal consists of two steps (see Figure 1.2):

- First, we analyze the special characteristics of geographic information that impose particular requirements over the design of the architecture of GIS applications.
- Then, we adapt the traditional three-tier architecture for general-purpose information systems to take into account the requirements extracted in the previous step.

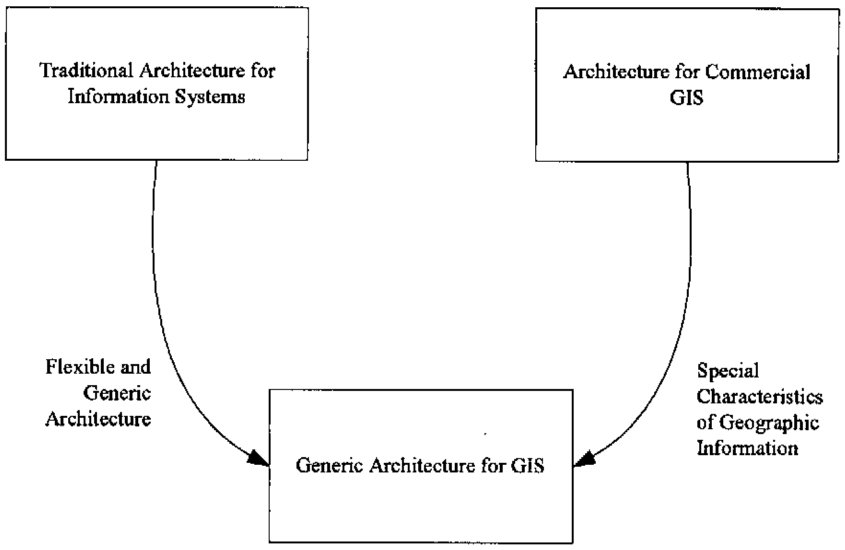

Traditional Architecture for Information Systems
Architecture for Commercial GIS
Flexible and Generic Architecture
Special Characteristics of Geographic Information
Generic Architecture for GIS

Figure 1.2: Towards a Generic Architecture for GIS.

This strategy enables us to design an architecture for GIS that keeps the benefits of the traditional three-tier architecture for general-purpose information systems while providing support for the special requirements of geographic information. GIS applications developed using this architecture inherit and enjoy important properties that are already present in the architecture of general-purpose information systems (e.g., interoperability, flexibility, generality, reusability, or scalability). The most important properties of the architecture proposed are its *flexibility* (i.e., the architecture can be adapted to support applications in many different technological environments), and its *generality* (i.e., the architecture can be used for multiple applications with different and specific purposes). At the same time, these GIS applications take into consideration the particular requirements of geographic information with respect to the architecture of the system. In order to achieve these goals, each component of the architecture must be described precisely, and illustrated by research results, existing standards, or commercial applications, as appropriate.

There have been many research and industrial efforts to standardize many aspects of GIS technology, particularly by the OpenGIS Consortium (OGC) and the ISO Technical Committee 211 (*ISO/TC 211, Geographic Information/Geomatics*). Their long-term goal is 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, [OGC99c, ISO02a]. These goals overlap with the main goal of our work, and therefore it is indispensable that we review thoroughly the OGC and the ISO proposals. We point out in this review the strengths of the standards and implementation specifications proposed by these organizations, and we also describe and analyze the drawbacks of these proposals. Finally, we describe the solutions adopted in our proposal to overcome these drawbacks while conforming with the standards where are available.

Nowadays, there are many commercial tools for the development of GIS that focus on different aspects of geographic information and computer technologies. These development tools have limitations that do not allow to implement well-structured applications that respond to the special requirements for geographic information, and provide important functional characteristics of information systems (e.g., flexibility, reusability or technology independence). In order to present these deficiencies, we have applied widely-used commercial tools in the analysis, design and implementation of a complete and complex real-life GIS, which is currently in use by the Provincial Council of A Coruña (Spain). We analyze the differences of this GIS with respect to our generic architecture that were imposed by the commercial systems used. This analysis allows us to point out the limitations that current commercial development tools have to design GIS applications with the desirable properties that the architecture we propose guarantees.

1.3 Thesis Outline

In this section, we describe the structure of this document. First, in Chapter 2 we analyze the special features of geographic information that impose requirements on the architecture of GIS beyond those common to general-purpose information systems. These features involve all levels of the information system, from data management to data presentation.

Then, in Chapter 3, we review the OGC and ISO proposals of standards for representing and manipulating geographic information. We also point out in this review the benefits and drawbacks of these proposals.

In Chapter 4, we present first a summary of the requirements for the architecture of general-purpose information systems, and we enumerate the requirements for the architecture of a GIS, which are extracted from the special characteristics of geographic information described in Chapter 2. Then, we introduce our proposal for a generic architecture for GIS that fulfills these requirements, and we present a detailed description of the components in the architecture and their interactions. Finally, we also compare our architecture to the OGC proposal and we analyze the improvements that our approach enables.

In order to show our experience in the development of a GIS that follows our generic architecture proposal, we describe in Chapter 5 the development of a GIS for the Provincial Council of A Coruña. We first give the background and motivation for the development of this GIS, and then we present the application schema and the workflow that was followed to build the GIS. Finally, we show the applications that were developed for the GIS, paying special attention to the web-based data exploitation tool.

Then, Chapter 6 is devoted to the evaluation of the differences between the proposed architecture and the implemented GIS application. These differences are caused by the limitation of commercial GIS development tools, and we analyze the consequences of these limitations.

We end this work by giving some concluding remarks and describing future work in Chapter 7.

Chapter 2

Special Characteristics of Geographic Information

2.1 Introduction

GIS researchers and practitioners have always assumed the existence of differences between general-purpose information systems and geographic information systems. However, we believe that it is necessary to analyze in a detailed way the special requirements that geographic information imposes on the architecture of general-purpose information systems in order to define a generic architecture for GIS applications that takes these requirements into account.

This chapter presents the first contribution of this work: a systematic analysis of the different aspects where GIS applications have different needs with respect to general-purpose information systems.

The rest of this section is structured as follows: Section 2.2 introduces some basic concepts, Section 2.3 describes how geographic information can be represented in a computer system, Section 2.4 describes the functionality that is needed to perform transformations, manipulations and analysis techniques, Section 2.5 gives detail on the process of presenting geographic information to the end-user, Section 2.6 analyzes the requirements that influence the architecture of GIS applications, Section 2.7 describes the importance of metadata for geographic information, Section 2.8 presents how GIS needs to be customized to support special applications, and finally, Section 2.9 presents a currently active research area that tries to integrate temporal and spatial functionality in a single system.

2.2 Basic Concepts

Geographic information has been an important and active research field since the early times of computer science because a lot of human activities and decisions involve a geographic component. This research area has involved much effort because geographic information has many particular characteristics that make it different from traditional business information. As an example, whereas the information of an employee in a traditional information system can be represented with simple values like numbers and integers, the geographic information associated to a meteorological model is much more complex. In order to provide efficient storage and manipulation of geographic information, the architecture of the information system must be adapted to accommodate the special nature and exclusive characteristics of geographic information and the requirements that emerge from the functionality required to the system.

In this chapter, we analyze the characteristics of geographic information which are peculiar to it, and we describe the requirements that they impose on the design of the architecture of a GIS. The first point of view that can be used to discover these requirements consists of an analysis of the functionality that must be provided by any GIS application. This functionality can be classified as follows (according to [BM98]):

- *Data input and verification.* This concerns all aspects of capturing geographic data, verifying their correctness and converting them to a common digital form.
- *Data storage and management.* It covers the structure and organization of geographic information both in terms of the way in which it is perceived by the user (i.e., *conceptual model*) and the way in which it is handled in the computer (i.e., *logical model* and *physical model*).
- *Data transformation and analysis.* This functionality consists of the processes of editing the information to keep it up to date or to remove errors. Data analysis is one of the main tasks of GIS, and concerns the application of analysis methods to the information to achieve answers to the questions asked to the GIS.
- *Data output and presentation.* The functionality of producing maps and map-based material is a highly distinctive feature of GIS compared with a general-purpose information system. Together with the analysis techniques, this is the aspect that differs the most from traditional information systems.

The cost of data input and verification is regarded as a major problem in the development of a GIS when the data do not exist or they are not suitable for the application. If the data do not exist, the system must provide tools to capture geographic information from existing paper or digital maps, field observations or sensors (including aerial photography, satellites, and recording instruments) and store it in the data management system. Furthermore, if the data already exist, they may be in a different format or referenced to a different reference system. Therefore, the system must include procedures to integrate these data in the data model of the application. Finally, it is necessary that the system provides methods and tools to check for correctness, integrity and consistency of any new data before they enter the system.

Regarding data storage and management, traditional information systems are not suitable for representing geographic information. Conceptual models for general-purpose information systems (e.g., the entity-relationship model) do not have constructs to model application schemas that deal with geographic information. Furthermore, logical models (e.g., the relational model) are strongly geared toward business applications that manipulate large but simple data sets, and do not include functionality to represent geographic information. The data types of logical models for traditional information systems are suitable for representing simple information such as numbers, texts, or dates. However, the representation of geographic information on a computer requires new data types such as *point, curve* or *surface*, or collection types such as *point collection, line collection, surface collection* or *geometry collection.* Similarly, the operations provided by logical models for traditional information system manipulate simple values. New operations are required to manipulate geographic information (e.g., *distance*, *direction*, or *intersection*). Finally, physical models for traditional information systems are unable to represent efficiently geographic information. Summarizing, traditional information systems must also be extended at all levels from the conceptual model to the physical model to represent geographic information adequately. Section 2.3 describes these issues in more detail.

Considering data transformation and analysis, there is a rich set of analysis techniques applicable to geographic information that require special transformation and analysis procedures (e.g., *path-finding algorithms in topological spaces*, *terrain analysis*, or *geographic data interpolation*). These techniques must be supported by primitive operations defined on the data types that represent geographic information, for instance: geometric operations (e.g., *distance*, *bearing*), topological operations (e.g., *adjacency*, *connectivity*) or set-oriented operations (e.g., *intersection*, *difference*). Furthermore, computational efficiency is an important requirement for these analysis techniques. This implies that efficient physical representations for geographic data, efficient access structures for fast data retrieval, and new algorithms for query processing must be defined. Section 2.4 provides more detail on these aspects of geographic information.

Finally, the characteristics of geographic information that make the process of presenting geographic information to a final user so distinctive in GIS applications are presented in Section 2.5. However, we can summarize these characteristics here as follows:

- *Geographic information requires special transformation and analysis techniques.* As was described above, geographic information requires special transformation and analysis procedures. Therefore, the user-interface of the GIS application must be prepared to enable the user to invoke these procedures and to visualize their results. For instance, the user interface must provide tools to draw geographic values to be used as parameters for queries.

- *Geographic information needs different visual representations according to different display parameters.* Unlike values from traditional data types like *number* or *string*, a value from a geographic data type is complex and it must be displayed with appropriate detail at different scales. For instance, the border of a city may be represented as a complex polygon at a large scale map, as a polygon with less detail at a medium scale map because the resolution of the display device may not allow to display all the detail of the complex polygon, and as a point at a small scale map.

In addition to the requirements that emerge from having to implement this functionality, there are other requirements that arise from the special nature of geographic information:

- *Geographic information is characterized by its large volume and its intrinsic complex structure.* This has two important consequences:
  - *The large volume of geographic data values must be taken into account in the design of the architecture of GIS applications.* Efficient data manipulation and transmission procedures between the components of the architecture must be designed and implemented. Moreover, the architecture of GIS applications is also largely affected by the need for new conceptual and logical models for geographic information. These issues are described in more detail in Section 2.6.
  - *The volume and complexity of geographic information causes metadata to be more important than in general-purpose information systems.* Appropriate metadata for geographic information is essential for its efficient use by end-users and applications. Section 2.7 is devoted to this topic.
- *The different types of analysis that may be performed by end-users are not limited.* No logical model for GIS applications can aim at providing a complete set of analysis operations. Therefore, the system must provide some primitive functionality that covers as much ground as possible, and offer the possibility to be extended by developers to implement advanced specific functionality (i.e., *customization*). This topic is further described in Section 2.8.
- *The time component of geographic information has often been neglected in research and commercial tools.* However, research and tools supporting dynamic geographic information are very important and necessary for some application

domains such as meteorology or fleet management. This has changed in the last years and a new research area has emerged named *spatio-temporal databases*. These research results are slowly being added to commercial systems. Section 2.9 briefly reports on these advances.

2.3 Geographic Information Management

2.3.1 Introduction

The phenomena that can be observed in the real world cannot be represented as such in a computer because the real world is infinitely complex and the storage space and the representation power of a computer are limited. Therefore, it is necessary to define *models* to represent real world information in a form that is suitable for a computer. A model is an artificial construction in which parts of a domain (i.e., the *source domain*) are represented in another domain (i.e., the *target domain*). A model allows developers to simplify and to abstract from the source domain by representing its elements with elements of the target domain. Processes that take place in the source domain can also be represented with operations on elements on the target domain. This allows the user of the model to simulate these processes and predict their outcome by applying operations on elements of the target domain and interpreting the results in the source domain. For the particular case of geographic information, the source domain consists of real world geographic phenomena, and the target domain consists of data elements that are stored in a computer. Manipulation algorithms and visualization techniques are applied upon the data on the computer, and the results are then interpreted as real world geographic phenomena (see Figure 2.1).

Two models are used in the development of a general-purpose information system: a *conceptual model* and a *logical model* (see Figure 2.2). A conceptual model (also called *abstract model*) is used to classify, identify and represent the phenomena of the real world in order to communicate it to other users or to the computer. However, a conceptual model cannot directly be represented in a computer system because it does not take into account the representation limitations of computer systems. For instance, geographic values are considered as infinite point sets with infinite resolution in the conceptual model, which cannot directly be represented using the limited precision of computer arithmetic. In order to represent phenomena of the real world in a computer, it is necessary to build a *logical model* (also called *discrete model*) that adapts the conceptual model to take into account the representation limitations of computer systems. Continuing with the same example, a logical model for geographic information represents the geographic values of the conceptual model using linear approximations of curve segments and surface borders.

The conceptual model more often used for the development of general-purpose information systems is the entity-relationship model [Che76]. This conceptual model

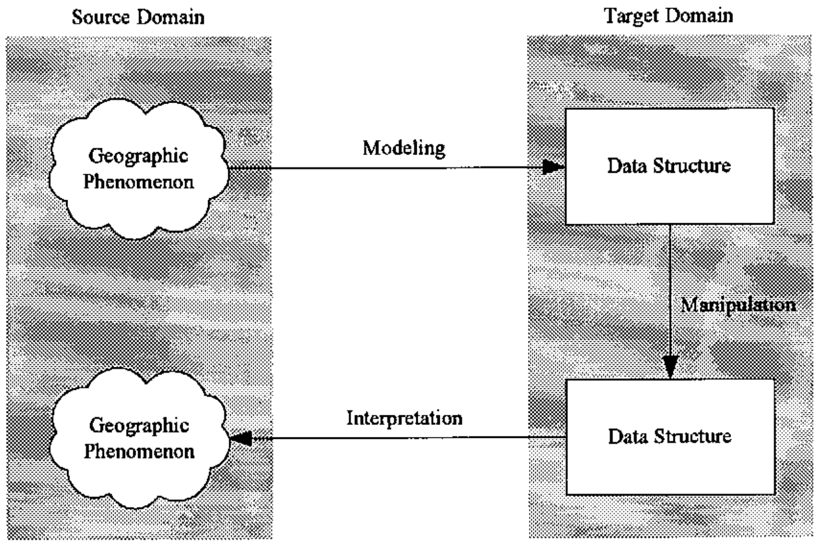


Figure 2.1: The Modeling Process.

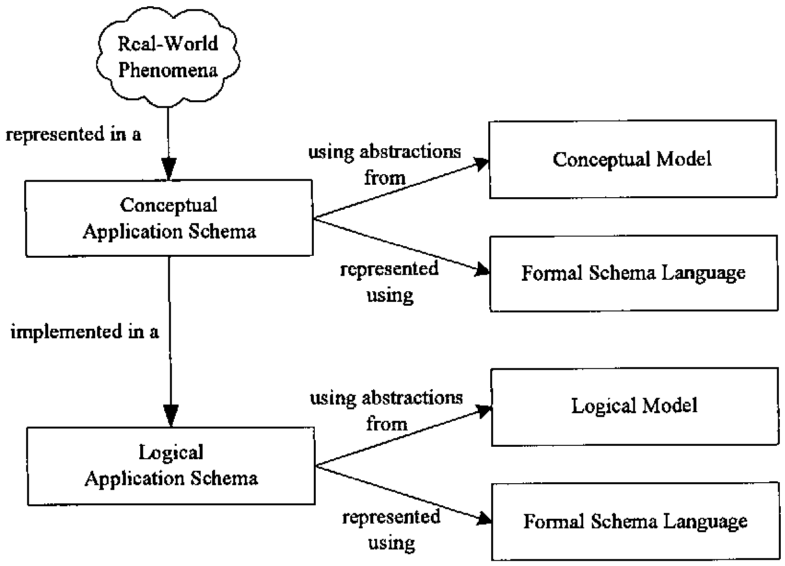


Figure 2.2: Models in an Information System.

is nowadays being replaced by the model defined by the Unified Modeling Language (UML) [BRJ98]. The logical model used to represent these models is often the relational model [Cod70]. Nonetheless, object-oriented models have also been used in domains where relational DBMS were not appropriate [Ban88]. Nowadays, the relational model has been extended to incorporate concepts from object-oriented models such as behaviour associated to the basic data abstractions [SM96].

These models cannot be used to represent geographic information because they are strongly geared towards business applications. New conceptual models and logical models are necessary. This has been a central issue in the research of GIS for many years [Güt94, SCR+99]. In the following sections, we describe how geographic information can be represented in a computer system. First, we describe in Section 2.3.2 the characteristics required by a conceptual model for geographic information. Then, the characteristics of a logical model for geographic information are analyzed in Section 2.3.3.

2.3.2 A Conceptual Model for Geographic Information

A conceptual model for geographic information must capture two different aspects of reality. First, it must provide constructs to represent *spatial and geographic data values*. Given that the traditional data abstractions (e.g., *number*, *string*, or *date*) cannot efficiently be used to represent geographic information, it is necessary to define a model with data abstractions for the representation of the geographic component of real-world phenomena such as the course of a river, the temperature values at noon in a country, or the division of a country into parcels that are owned by people. This model is often called *geographic data model* or *spatial data model*.

Secondly, the conceptual model must provide constructs to describe the data schema of an application that manipulates geographic information. This is often referred to as the *geographic application schema*. We will describe the requirements of each part of the conceptual model separately.

Conceptual Representation of Geographic Values

The conceptual model for geographic values must provide data abstractions for the representation of *geographic space*. It was recognized at an early stage in the development of GIS technology that geographic space may be conceptualized in two distinct ways [Chr75, Chr78]: space can be considered as an element existent in itself that has properties associated to each location, or it can be considered as a container for objects with properties associated. Each point of view has originated a different family of spatial data models, namely *field-based models* and *object-based models*.

Object-based models treat the information space as populated by discrete, identifiable entities, each consisting of geographical shape and a set of descriptive

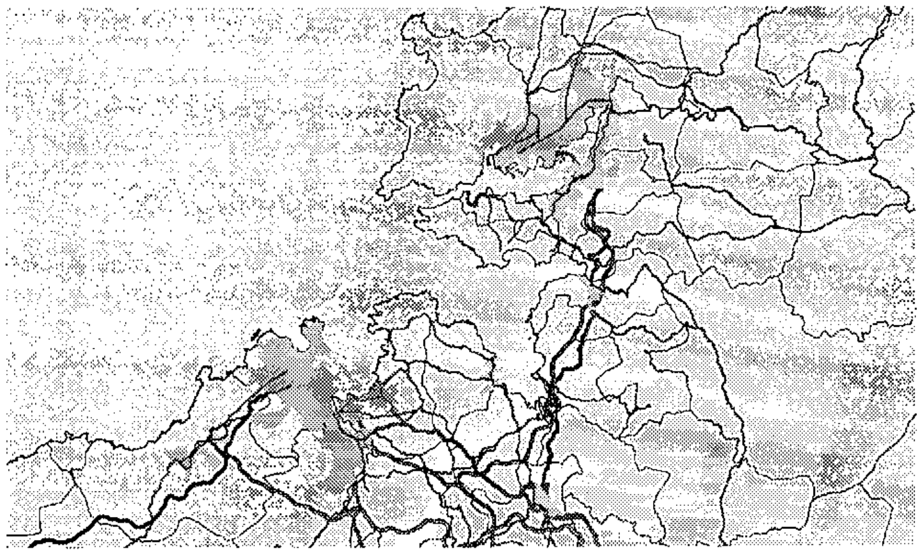


Figure 2.3: Object-based Geographic Information.

properties. Figure 2.3 shows an example of object-based geographic information. This figure shows a map of the Spanish province of A Coruña that depicts the municipalities and the main roads. Each municipality is manipulated and drawn as an independent polygon, and each road is manipulated and drawn as an independent line. This view of geographic information gathers within a spatial object points of the geographic space that share several common properties (i.e., having the same set of descriptive attribute values). In order to distinguish an object from others, an explicit identity is assigned to it.

On the other hand, field-based models treat geographic information as collections of spatial distributions, where each distribution may be formalized as a mathematical function from a spatial framework to an attribute domain. Figure 2.4 shows field-based geographic information that was added to the information in Figure 2.3. The colored areas represent the amount of rain expected at every location at a particular time instant. In a field-based model, one or several attribute values are associated to each point in space. In our figure, this attribute value is the expected amount of rain. Identifiable objects and zones are only recognized when there are remarkable clusters of similar attribute values or significant events in space, time, or attribute values.

Field-based and object-based models are in a sense inverse to each other. In field-based models, geographic information is represented as a function from space to a set of attribute values. On the other hand, in object-based models, geographic information is represented as entities that map a set of attribute values to a set of points in space. An analysis of both families of models can be found in [Cou92].

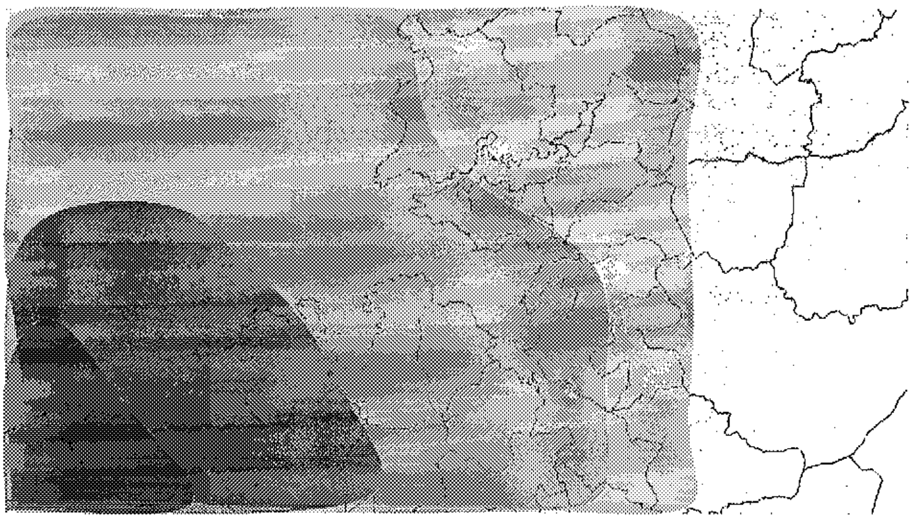


Figure 2.4: Field-based Geographic Information.

Both models are needed in a GIS because each of them is better suited for a different set of analysis techniques. As an example, the analysis of geographic information collected by sensors (e.g., satellites or meteorological stations) is better performed using a field-based model. On the other hand, object-based models are more suitable for the analysis of man-made structures and divisions (e.g., road networks, administrative divisions, cadastre).

In order to be included in a conceptual model for GIS application schemas, both kinds of models define *domains* or *data abstraction* that are later used as part of the conceptual model for the GIS application schema.

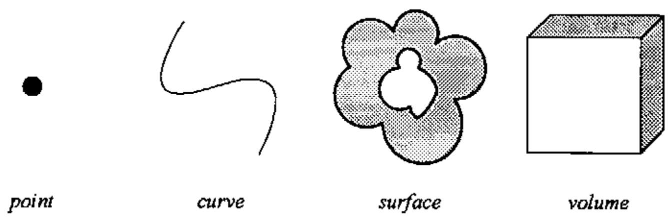


Figure 2.5: Examples of Object-based Data Abstractions.

Object-based models usually define a set of data abstractions to represent the different kinds of geographic objects supported by the model. The choice of data abstractions is usually based on the dimensions of the geographic object, for instance *point* (0-dimensional), *curve* (1-dimensional), *surface* (2-dimensional) and *volume* (3-

dimensional). Figure 2.5 shows example values of these data abstractions. In addition to these, data abstractions to represent homogeneous and heterogeneous sets of objects are often defined. For instance, *point collection*, *curve collection surface collection*, *volume collection* and *geometry collection* (see Figure 2.6 for example values of these data abstractions). Finally, abstractions that represent restricted cases of the general data abstractions are usually defined as well. As an example, consider the data abstraction *line ring* that represents a closed line.

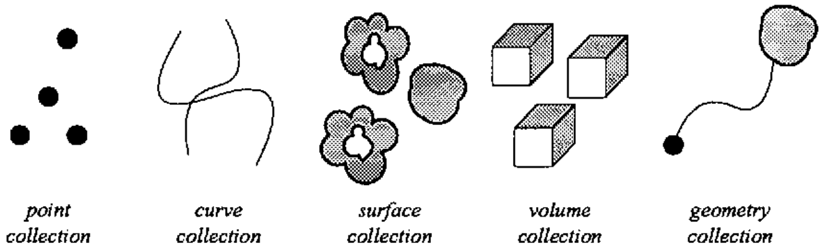


Figure 2.6: More Examples of Object-based Data Abstractions.

On the other hand, field-based models usually define a single data abstraction that represents the field function. The name most commonly assigned to this data abstraction is *coverage*, though other names such as *partition* are also used.

Conceptual Model for GIS Application Schemas

In addition to a conceptual model for representing the geographic component of real-world phenomena, it is necessary to define a conceptual model that enables the developer to define the schema of any GIS application. Many conceptual models for geographic information have been proposed in the literature, and it is outside the scope of our work to provide a detailed description of them. A thorough survey can be found in [Viq03].

A common characteristic of all conceptual models is that they provide an abstraction for real world phenomena. In the entity-relationship model [Che76], this abstraction is called *entity*. In object-oriented models [RBP$^+$91], this abstraction is called *object*. In the particular case of conceptual models for geographic information, many names have been used (e.g. *geographic object*, *spatial object*, *feature*). We will use the term *feature* in the following descriptions.

Each conceptual model proposed defines an abstraction that captures different aspects of the real world. However, it is nowadays generally accepted that the following aspects of real-world phenomena must be represented by this abstraction:

- *Information regarding properties of the real-world phenomenon.* This information is represented as a set of *attributes* that may be:

  - *Geographic attributes.* These attributes describe the geographic characteristics of the feature. Alternative names are *spatial attributes* or *geometric attributes.*
  - *Descriptive attributes* These attributes represent the characteristics that are unrelated to the geographic component. They are also called *alphanumeric attributes.*

- *Relationships between the real-world phenomenon and other real-world phenomena in the model.* It consists of associations between a feature and other features (or itself). An example of this type of information is a topological association that describes an adjacency relationship.

- *Behavioral aspects of the real-world phenomenon.* This behaviour is represented in the feature as a set of operations that can be applied upon the feature to perform analysis procedures.

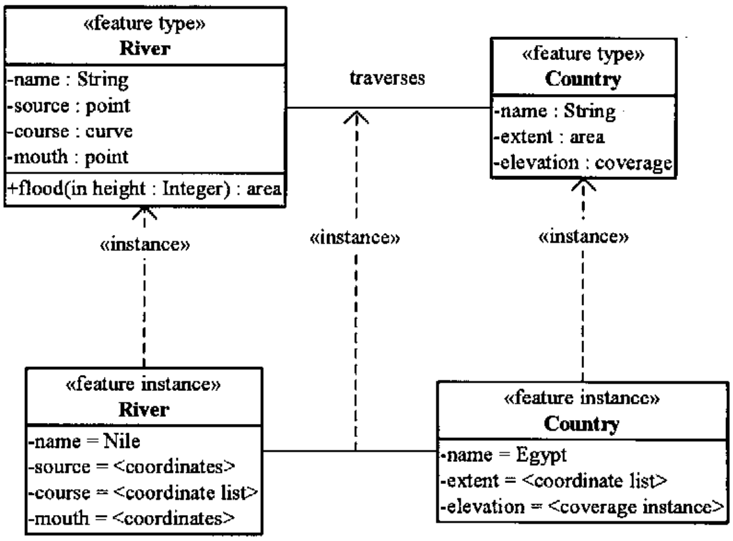


Figure 2.7: An Example Representation of a Conceptual Model.

Consider, for instance, the real-world phenomenon *a river* with the properties: *name, location where it has its source, course,* and *location where it flows into the sea or into another river.* Furthermore, it is interesting for the GIS application to compute the flood caused by a raise in the river level of a given number of meters. Using abstractions from the conceptual model that we have just defined, this real-world phenomenon can be represented using the application schema in Figure 2.7 (expressed using UML). The real world phenomenon is represented as a feature with an alphanumeric attribute that represents the river name, a geographic attribute of type *point* that represents the location where the river has its source, a geographic attribute of type *curve* that represents the river course, and a geographic attribute of type *point* that represents the location where the river flows into another river or the sea. The feature representing the river is also associated to the features in the application schema that represent the administrative division of the country by the relationship *traverses*. Finally, the behaviour of the river when a flood occurs is represented by the operation *flood* that simulates a flooding by raising the water level of the river and computing the areas of the country that are affected using the relief map of the country represented with a geographic attribute of type *coverage*.

This example shows how the conceptual model is used to represent geographic information. Real-world phenomena (e.g., a river, or a country) are represented by features. The information that characterizes the phenomena is represented by properties of the features (e.g., the river name, or the country name). Furthermore, data abstractions such as *point* or *curve* are used to represent geographic properties of the features using an object-based view of the geographic space (e.g., the river source and mouth, or the river course). Finally, a *specialized feature* (in the object-oriented sense of the term) denominated *coverage* is used to represent geographic properties of the features from a field-based view of geographic space.

A representation of the real world in this conceptual model consists of set of individual features. These features are called *feature instances*, and are akin to entities in the entity-relationship model or objects in the object-oriented model. In order to classify the real world phenomena into classes that are used to describe the schema of a particular application, individual features are grouped into classes with common characteristics. These classes are called *feature types*, and are akin to entity types in the entity-relationship model or object types in the object-oriented model.

Finally, an abstraction is needed to represent an homogeneous collection of feature instances (i.e., having the same feature type and the same semantics). This collection is used to gather all information corresponding to a particular topic in order to be manipulated as a set. This abstraction is commonly called *feature collection*, *geographic theme*, *geographic layer*, *thematic layer*, or simply *layer*.

In order to model an application domain using this conceptual model, a developer analyzes the requirements of the application and identifies the types of features that are needed. This is done by defining an application schema consisting of a set of *feature types*. Each feature type consists of the specification of the attributes, operations and associations that are used to represent a particular collection of real-world phenomena. Then, the application schema is filled with *feature instances* that represent each a real world phenomenon.

2.3.3 A Logical Model for Geographic Information

The previous section has informally defined a conceptual model for geographic information, by defining the data abstractions that are used in the model to represent geographic information. This conceptual model is *abstract* in the sense that no details have been given regarding the representation of the model abstractions in a computer system. For instance, geographic values are represented in the conceptual models as infinite point sets with infinite precision. This cannot directly be represented in a computer system because the storage space and the arithmetic precision are limited. Therefore, it is necessary to define a logical model for the representation of geographic information that overcomes the representation limitations of a computer system.

This section describes the main characteristics of logical models for geographic information. We will describe first a logical model for the representation of geographic data values. Then, we will enumerate some considerations that must be taken into account when the logical model is designed.

Representation of Geographic Data Values

There are two different approaches for the representation of geographic information in a computer: the *tessellation model* and the *vector model.*

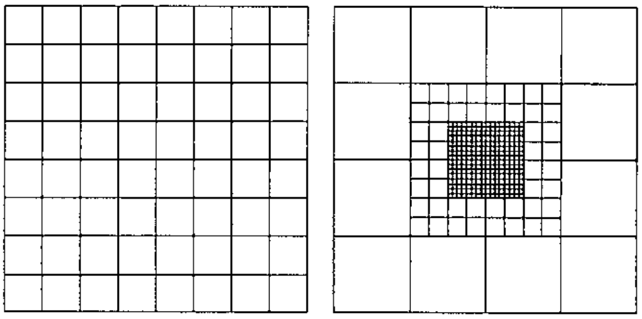

(a) Fixed Tessellation Model (Raster). (b) Variable Tessellation Model.

Figure 2.8: Tessellation Model.

In the tessellation model, the geographic space is decomposed into an array or grid of disjoint cells and each cell is assigned a set of attribute values. This approach can be further divided into *fixed* (or *regular*) and *variable* (or *irregular*) tessellation models. A fixed tessellation model uses a regular grid or *raster*, which is a collection of units of equal size and shape. In the raster model, each cell is addressed by its position in the array (row and column number), and cells are often referred to as *pixels*.

A variable tessellation model handles units of decomposition of various sizes and shapes. Geographic values are represented as collections of cells from the tessellation. Figure 2.8 presents examples of a fixed tessellation and a variable tessellation. The figure on the left shows a fixed tessellation of space using a regular grid of square cells. The figure on the right depicts a variable tessellation of space that assigns more cells to the central part of the space in order to capture more detail.

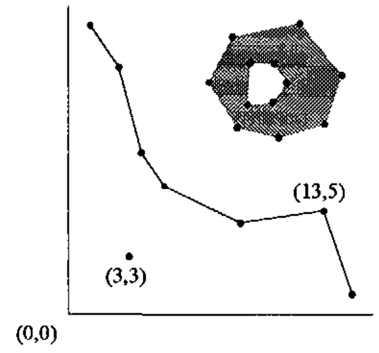


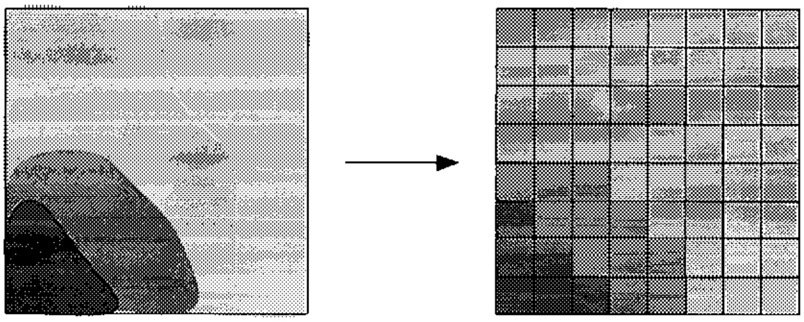
Figure 2.9: Vector Model.

In the vector model, geographic information is represented by points and edges. A point is represented by its coordinates with respect to some reference system of the space, and an edge is represented by a finite straight line segment defined by its endpoints. Figure 2.9 shows some geographic values represented using a vector model. The axes of the coordinate reference system are shown as well as the coordinates of some points in the objects. A discretization of geographic space is not explicit in the vector model as it is with the tessellation model. However, it must exist implicitly in some form, because of the discrete nature of computer arithmetic.

The choice of a conceptual model and a logical model are orthogonal because any logical model can be used to represent any conceptual model. A field-based model can be represented using both the tessellation model or the vector model, and an object-based model can be represented using the vector model or the tessellation model.

Figure 2.10(a) shows how a field-based model can be represented using the tessellation model. Field-based information is still represented as a function from space to a range. However, the function domain is no longer the infinite set of points, but a finite set of cells. Space is no longer seen as a continuous field, but as a discrete one, which allows for an explicit representation of data values.

Furthermore, a field-based model can also be represented using a vector model. The function from space into a collection of attribute values is no longer represented at all points in space. Instead, a finite collection of sample values are used, and the values of the other points are obtained by interpolation (see Figure 2.10(b)). A common type


(a) Tessellation model.

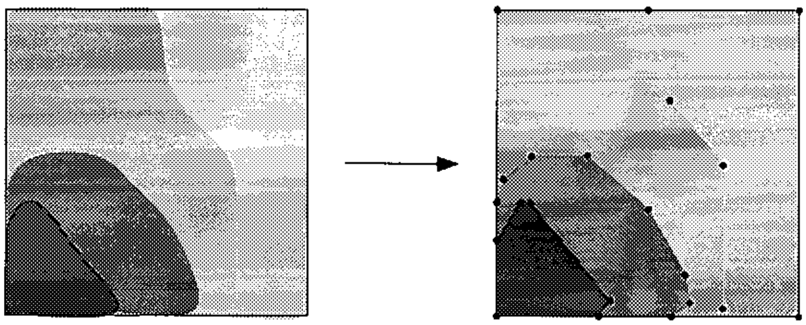
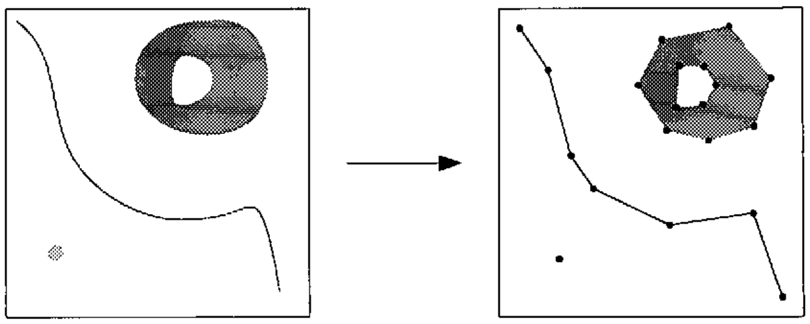
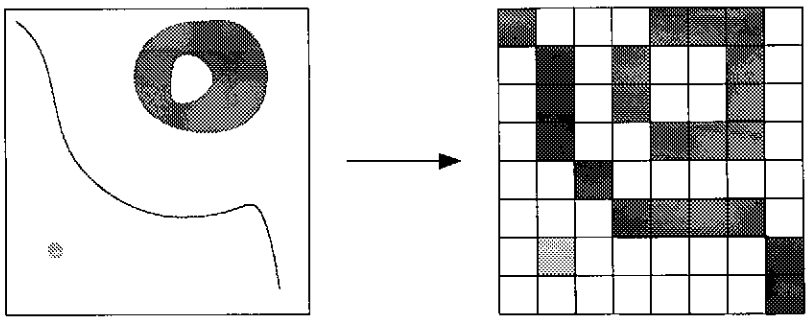

(b) Vector model.

Figure 2.10: Representation of a Field-based Model.

of representation are ***triangulated irregular networks (TIN)***, that represent space using a triangular partition where no assumption is made on the distribution and location of the vertices of the triangles. The attribute values are recorded at each vertex, and inferred at any other point by linear interpolation of the three vertices of the triangle that contains the point .



(a) Vector model.



(b) Tessellation model.

Figure 2.11: Representation of an Object-based Model.

A vector model can be used to represent an object-based model (see (Figure 2.11(a)). A point is simply given as a coordinate. An arc is discretized as a sequence of straight-line segments, each represented by a vector, and a surface is defined in terms of its boundary, represented by a collection of vectors. There exists a large number of variants to represent arcs and surfaces in vector model.

Finally, a tessellation model can also be used to represent an object-based model (see Figure 2.11(b)). Each spatial object is represented by the smallest set of cells that contains it. This set is finite, and therefore, representable. Thus, a point may be represented as a single cell, an arc by a sequence of neighboring cells and a connected area by a collection of contiguous cells.

Many different vector and tessellation models have been proposed in the literature, and it is outside the scope of our work to provide a detailed description of each model and a comparison between the different models. However, we can briefly compare and summarize the advantages and drawbacks of each kind of model with respect to the other. We can affirm that generally, the vector model is more suitable for the representation of object-based models and the tessellation model is more appropriate for the representation of field-based models. However, the following aspects influence the decision [BM98, LGMR01, LT92, RSV01, Wor95]:

- *Representation efficiency.* The vector data model is more efficient in the use of computer storage than the tessellation model because only points of interest have to be stored. On the other hand, the tessellation model implies large data volumes. The larger the grid resolution (i.e., the smaller the cell size), the better the approximation and the higher the number of cells needed to represent the information. This implies that the representation occupies much memory space and operations on objects are more time-consuming. A tessellation stored in a raw state with no compression can be extremely inefficient in terms of computer storage usage.

- *Computational efficiency.* The vector model implies the storage and manipulation of complex data structures (lists of points and edges). Some operations (e.g., intersection of geographic values, overlay of feature collections) are complex and require considerable computing power. On the other hand, the tessellation model uses simple data structures that are handled naturally by computers because all commonly used programming languages support array handling and operations. This implies that spatial analysis and filtering is easy.

- *Expressive power.* The vector model allows the explicit representation of topological relationships between geographic objects, and therefore can easily be used to perform topological analysis such as path-finding in networks. The tessellation model is not suitable for this kind of analysis. On the other hand, the vector model assumes a that real-world phenomena have crisp boundaries, which does not agree with our observations. To change this viewpoint, another level of complexity must be added to the representation (e.g., using fuzzy coordinates for the end-points of the edges).

- *Supported transformations.* Coordinate transformations, affine transformations (e.g., rotation, translation) and adjustments to surface models are simple in vector models. However, this transformations are time-consuming, and need complex algorithms in tessellation models, which may result in loss of information and distortion.

- *Location-specific analysis.* The manipulation of alphanumeric data for each location of geographic space is simpler in the tessellation model than in the vector model. The process of retrieving the value of an attribute at a specific location of geographic space is a simple task in the tessellation model, because it only requires to retrieve the cell that represents that location, or to interpolate the values of the surrounding cells. On the other hand, the same analysis in the vector model requires to check all geographic objects to see if they contain the given location (i.e., a *point query*), which is a more complex task.

- *Data presentation.* Data displaying and plotting may be time-consuming and expensive in vector models, but the graphical representation is accurate at all scales. On the other hand, raw maps produced from tessellation models are inelegant but simple to produce.

- *Data production.* The tessellation model is of prime importance because of the exponentially growing volume of data coming from satellite sources and the its increasing use in scientific applications such as environmental fields (e.g., pollution, weather). The vector model is important for the huge amount of data coming from CAD sources.

Systems have tended to specialize in one of these models because each requires different technologies and each originates from different viewpoints of geographic information. Vector data models were seen as being truer to conventional cartography and were used for applications where high-quality map-making was required. On the other hand, tessellation models were seen as being suitable for overlay analysis and sensor-based information.

This specialization is no longer relevant because it has been shown that what seemed to be an important conceptual difference, it is in fact largely a question of technology. Today, many GIS development tools support both models and provide translation procedures between them (i.e., *rasterization* for vector to raster conversion and *vectorization* for raster to vector conversion).

Trade-offs in the Definition of a Logical Model

The definition of a logical model for geographic information involves deciding a number of data types to represent geographic data values. This decision involves in turn a number of trade-offs that have been the cause of the definition of such a high number of logical models for geographic information. It is outside the scope of our work to describe and analyze them. An exhaustive survey and comparison can be found in [Viq03]. Nevertheless, now we give a brief summary of the consequences of finding a compromise between the following pairs of characteristics:

- *The expressive power versus the simplicity of the type system.* A simple type system consist of few data types with clear semantics. This type system may be unable to model complex geographic phenomena. For instance, consider the real-world phenomenon depicted in 2.12(a) that consists of a river that has a dam in the middle of the course. If a simple type system consisting only of the data types *point*, *curve*, and *surface* is used to represent this real-world phenomenon, then more than one object must be used because the area of the dam cannot be represented as a line and the course of the river cannot appropriately be represented with a surface. The real-world phenomenon is a mixture of a line and a surface. On the other hand, this real-world phenomenon can be represented using a complex type system that has a type for heterogeneous collection of geographic values. For instance, type system defined by the ISO/TC 211 to represent geographic information in [ISO03b] includes dozens of data types. The problem with complex type systems is that it may not be easy to understand the difference between some of them.

Furthermore, if a simple type system is used, it may be impossible to define some operations because the result may not be representable in that type system. For instance, in the general case, the intersection between two geographic values of type *curve* is a set of points (the curves may intersect at more than one point). The simple type system described previously cannot represent the result of this operation using the data type *point*, and therefore, the operation cannot be included unless new data types are defined.

- *A weak type system versus a strong type system.* A *weak type system* defines data types that do not impose strict constraints on the values. This results in a type system that is more intuitive and closer to the user perception. On the other hand, a *strong type system* imposes rigid constraints on the data types that avoid incorrect or inconsistent geographic values but may be unable to model some geographic phenomena.

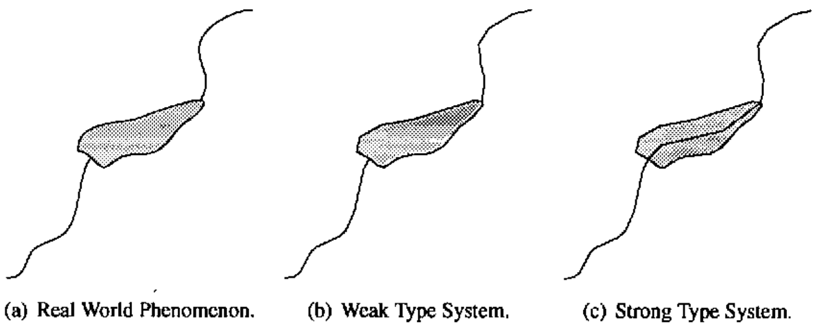

(a) Real World Phenomenon. (b) Weak Type System. (c) Strong Type System.

Figure 2.12: Weak Type Systems vs. Strong Type Systems.

As an example to illustrate the importance of this decision, consider how different the two classes of type systems model a real-world phenomenon consisting of a river that has a dam in its course (see Figure 2.12(a)). This can be represented using a curve that becomes a surface at the dam and then becomes a curve again after the dam. In a weak type system, this can be modeled using a geographic attribute of type *surface* that is *degenerated* into a curve when the river is outside the dam (see Figure 2.12(b)). A geographic attribute of a type that allows heterogeneous collections of spatial objects may be used as well. On the other hand, a strong type system considers this geographic object an invalid case, and it must be modeled as two independent objects, namely a river and a dam (see Figure 2.12(b)).

2.3.4 Physical Model

Whereas the data types of the logical model of traditional information systems (e.g., the relational model) can be stored using simple storage tools like *disk pages* and *records*, geographic information requires data types whose representations may vary in size from very small to very large. Consider, for example, the implementation of a *surface* data type represented by a list of vertices. A value of this data type could represent the border of the US state of Colorado (which is exactly a rectangle and would need exactly four pairs of coordinates) or the border of Norway (which has thousands of vertices because of the many fjords). Therefore, the data structure supporting this data type must be prepared to deal with values of different sizes.

Furthermore, these data values are generally used as components of a comprising structure (e.g. as attribute values in a tuple). If the storage size of the representation of a value is small, the best approach is storing it *inline* within a storage block representing a tuple in order to retrieve the value when the tuple is fetched and save a disk access. On the other hand, if the representation is large, it is better to store it using a separate BLOB, especially if the value is accessed only rarely. For a given *size* and *access probability* of a large object, query performance depends on its representation: either inlined within the aggregate or swapped out to a separate object.

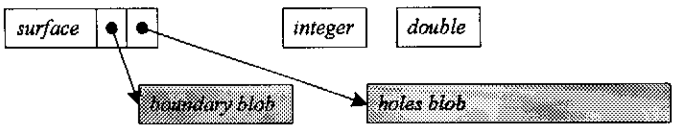


Figure 2.13: Storage Representation of Attribute Values.

Consider, for example, a geographic feature that represents a country. Each instance of this feature type consists of an attribute of the data type *integer* representing the population of the country, an attribute of the data type *double* representing the average height of the people in the country, and an attribute of the data type *surface* representing the geographic extent of the country. Furthermore, the storage representation of a *surface* value consists of a fixed-size record storing useful properties of the value, and two large objects storing a list of coordinates for the boundary of the surface, and a list of coordinates for the holes of the surface. The storage representation of these values is shown in Figure 2.13.

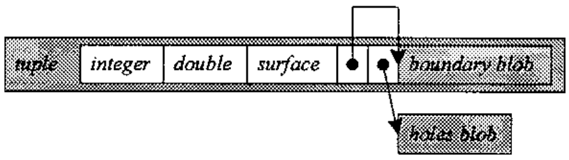


Figure 2.14: Storage Representation of a Disk Tuple.

Finally, when these values are embedded in a disk tuple, the best storage layout depends on the *size* and *access probability* of the large objects [DG00b]. Figure 2.14 shows an example layout for a tuple containing these values. In this particular case, the system has determined that the best layout is the one where the large object representing the boundary of the surface is placed inside the disk tuple with the other attributes of the feature instance, whereas the large object representing the holes of the surface is stored in a separate disk page.

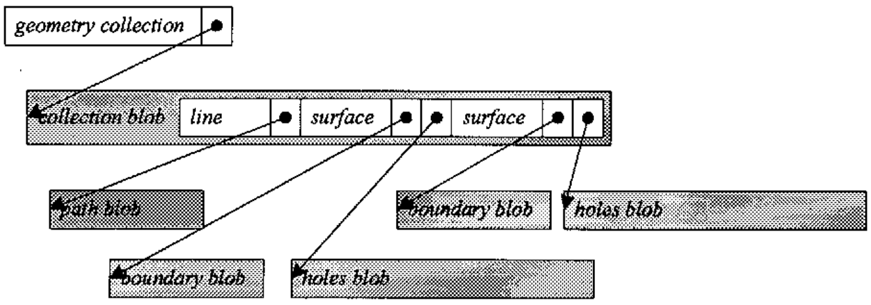


Figure 2.15: Storage Representation of a Geometry Collection.

Furthermore, the implementation of complex data models often requires *nested* large objects, and access performance is highly influenced by the clustering strategy followed to store the resulting tree of large objects. Consider, for instance, the storage representation of a *geometry collection*, as shown in Figure 2.15. In addition to a fixed-size record with properties of the value, the list of geometry values is stored in a large object. Given that the data type for each geometry value uses large objects for its representation, the result is that the storage representation of a *geometry collection* value is a tree of large objects.

We have presented in [DGL00] a large object extension that automatically clusters trees of nested objects. A sound and general interface was described, and two different clustering techniques using a cost model based on access probabilities and object sizes were given. A *rank function* is developed which indicates the suitability of a large object being inserted into a given cluster. In a series of simulations, we analyzed how the algorithms compare to each other and to two trivial ones. The results showed that they improve efficiency and that one outperforms the other in all situations. However, a detailed description of these tools is outside the scope of this work.

In addition to storage structures providing efficient physical representation of the geographic information [CG02], index data structures for the implementation of efficient access methods are needed. This has been a very active research area because spatial access structures are the basis of other application domains such as *terrain visualization* and *three-dimensional visualization*. More details regarding these techniques can be found in [Sam90, GG98].

2.3.5 Summary

We can summarize the requirements described in this section as follows:

- It is necessary to define a conceptual model that supports the description of geographic information (Section 2.3.2). This model must provide the following abstractions:
  - Abstraction for real world phenomena.
  - Abstractions to classify real world phenomena into classes that are used to describe the schema of a particular application.
  - Abstraction to represent collections of phenomena.
  - Abstractions to represent the object-based view of space.
  - Abstractions to represent the field-based view of space.
- Given that the conceptual model provides abstract construct for modeling geographic information, it is necessary to define a logical model (also called discrete model) that takes into account the limitations of computer systems with regard to the representation of information (Section 2.3.3). This model defines data structures for the data abstractions and operations that can be implemented in a computer system.
- Even though the logical model takes into account the limitations of computer systems, different technologies require different implementations of the logical model. Therefore, it is necessary to define a physical model consisting of data structures and algorithms that represents the logical model abstractions in a given technological environment (Section 2.3.4).

2.4 Geographic Information Processing

Section 2.3 sketches a conceptual model and logical model for the representation of geographic information in a computer system. The conceptual model defines abstractions to describe real-world phenomena, and the logical model defines data types for the representation of the conceptual model abstractions in a computer system Once this has been achieved, it is necessary to provide functionality to perform transformations, manipulations and analysis techniques on geographic information. This functionality can be used to add value to the geographic information, to support decisions, or to reveal patterns, anomalies and special features that are not immediately obvious. Whereas geographic data modeling is the process by which real world phenomena are converted into data that are represented in a computer, *geographic data analysis* is the process by which data are turned into useful information.

Geographic data types require special transformation and manipulation techniques that must be integrated in the GIS in order to capture the rich set of possible analysis techniques applicable to geographic information. Moreover, computational performance is an important problem for GIS because geographic data are notoriously voluminous and complex: efficient physical representations, access structures, and operation algorithms must be defined for geographic information. This section describes the special requirements that geographic analysis imposes on the architecture of GIS applications. First, we give in Section 2.4.1 a brief introduction to the different kinds of primitive operations that can be performed on geographic data. Then, we briefly describe in Section 2.4.2 some typical problems that can be efficiently solved by GIS using these primitive operations. Finally, we end this section presenting some concepts regarding physical representations, access structures, and algorithms in Section 2.4.3.

2.4.1 Geographic Operations

The purpose of this section is to provide a summary of the different kinds of basic operations on geographic information in order to illustrate the large variety and complexity of functional requirements. There have been many different attempts to define an exhaustive collection of operations. Our enumeration is not exhaustive, only categories of operations are given together with some examples of operations within each category. The enumeration is not minimal either because some of the operations can be expressed by others in the same list. Finally, the purpose of our categorization of the operations is merely illustrative, we do not claim that is a well-founded ground to design the set of operations in a GIS.

We have described in Section 2.3.2 the two alternative conceptual views of space, either as a container of geographic entities (*object-based* view of space) or as an entity existent in itself with attributes associated to each location (*field-based* view of space).

Each view of space requires a different set of operations. Furthermore, there are operations that are independent of the view of space because they can be defined for both conceptual views of space. We will explain these three types of operations in three separate sections.

Operations on Geographic Entities

There are many different ways in which operations on geographic entities of an object-based model can be classified. The most obvious is to classify the operations according to the type of result that is returned:

- *Operations with Boolean result.* These operations test a spatial object for a given property. Operations in this category are also called *spatial predicates*. The property checked may be inherent to the geographic entity (e.g., *isConvex*, or *hasHoles*), or it may be a relationship to another spatial objects (e.g., *isNorthOf*, *touches*, or *overlaps*).
- *Operations with scalar result.* These operations compute a numerical property of a spatial object. Examples of this category of operations include *length*, *area*, *bearing*, or *distance*.
- *Operations with spatial result.* These operations compute a new geographic entity derived from an existing object. Additional parameters may be necessary to express the desired particular computation. For instance, *translation*, *rotation intersection*, *union*, *difference*, *boundary*, or *centroid*.

Another categorization of operations on geographic entities, which is orthogonal to the previous one, classifies the operations according to the type of space in which the operation can be applied:

- *Operations on a point-set space.* These are operations that consider geographic entities as infinite point-sets, and perform set oriented computations on the geographic entities. Examples of such operations are *intersection*, *union*, *difference*, *isAMember*, or *isASubset*.
- *Operations on a topological space.* These operations are related to properties of the geographic entities that are preserved by *topological transformations* of space. Some operations in this category are *boundary*, *interior*, *meets*, *isEnclosedBy*, or *covers*.
- *Operations on metric spaces, particularly the Euclidean space.* These are operations that apply on spaces where the *distance* property has been defined using a specific metric. Some operations that belong to this category are *distance*, *area*, *bearing*, *isNorthOf* or *centroid*.

- *Other operations.* There are some operations that are independent from any particular kind of space. For instance, the operation *equals*, which can be defined on any of these spaces.

Finally, the relational model for data management has established an important precedent in the definition of algebras because it defines a small set of operations that enable a great number of data manipulation processes (e.g., projection, selection, join). The combination of these operations with predicates and functions previously defined must be provided within the functionality of any GIS development tool. For instance, it must be possible to apply a *selection* or a *join* using a geographic predicate as the condition.

Operations on Geographic Fields

Operations on field-based models cannot be classified using the same approaches as operations on object-based models because space is now considered as an entity with existence in itself, instead of a mere framework for other entities. Given that geographic space is modeled using a function from space to an attribute domain (i.e., a *field function*), operations on field-based models take as arguments field functions and produce a new field function that results from the computation of the operation. Therefore, these operations constitute an *algebra* on field functions (also called *map algebra*). A categorization of these operations has been given in [Tom90], which is the basis for the map algebra definition in many commercial GIS development tools. The operations are classified in the following groups:

- *Local operations.* In these operations, the value of the resulting field function at any given location is computed as a function of the values of the argument field functions at the same locations. An alternative name is *point operations.* Examples of this category are *sum*, *difference*, or *mean*.

- *Focal operations.* The value of the new field function at a location is result not only of the values of the argument field functions at that location, but also of the values in the neighborhood of the location. A good example is the operation *slope* that takes a field function representing the elevation at each location, and computes the maximum gradient of the slope at each location by comparing the elevation at one location with the elevation values in the neighboring locations. Another example of this category is the operation *aspect* that computes the bearing of the maximum slope at each location.

- *Zonal operations.* Whereas focal operations compute the value of the resulting field using a neighborhood of values defined similarly for each location, zonal operations aggregate values of a field using the zones defined by another field. In order to compute the resulting value for a location, first the zone to which it belongs is determined, then the values of the argument function fields are collected, and finally the operation is applied to this collection of values. An example is the operation *zonalmean* that can be used to compute the average temperature in each administrative division on a country using a field function that represents the temperature values and the zones defined by another field function that represents the partition of the country into administrative divisions.

Operations on Collections of Geographic Objects

The operations presented in the previous sections were each specific of a particular conceptual view of space. However, there are geographic operations that are not specific of any conceptual view because they can be defined easily for both views. We present here some of these operations:

- *Overlay.* The most common operation on collections of geographic objects is the *overlay* operation. The result of this operation applied to two collections of geographic objects is intersection of the geographic attributes and the combination of the descriptive attributes of the objects in the collections.
- *Buffering.* The formation of areas containing locations within a given range of a given set of features.
- *Interpolation and extrapolation.* In situations where the available information is limited, it is often necessary to estimate the value of a given attribute at a position where no information is available. This process is called *interpolation* when the new values lies within the range of the available information, and *extrapolation* when it is applied to find a value in an area where no other value is present.

2.4.2 Geographic Problem-solving

The operations described in the previous section provide basic functionality over geographic information. In order to solve real-life problems with this functionality, the operations must be combined into problem-solving techniques. This section describes some of the typical techniques for problem solving in GIS tools.

We can categorize the problem-solving techniques that must be offered in a GIS development tool by defining the questions that these techniques must help answering:

- *What is it at this location?, and Where is this located?*
- *What is the spatial relationship between these objects?*
- *What location satisfies these requirements?*

- *What will be the situation in the future?*

The first and most important type of problem that must be solved by GIS tools involves answering the questions *What is it at this location?* and *Where is this located?* The answer to the first question is the geographic objects or attribute values at a given geographic location, whereas the answer to the second question is the location at which some particular conditions hold.

These queries represent opposite ways of accessing the information in the system. Using the first type of query, the user gives geographic information to the system in the form of a location and retrieves the descriptive components of the objects found. With the second type of query, the user gives descriptive information of the objects to be retrieved and the result is the geographic component of the objects. GIS development tools must provide mechanisms to perform both types of queries using a query language that allows to express both traditional queries and this new type of queries.

A second type of problem that must be solved by GIS applications is the computation of properties of geographic information. This problem can be briefly described as answering the question *What is the spatial relationship between these objects?*

Regarding geographic entities, this question is answered by combining the techniques for the previous problems with the application of the operations described above. That is, first, the entities must be selected, and then the fulfillment of a geographic predicate is evaluated on the selected objects. Proximity analysis is an illustrating example of this type of problem, which involves finding the geographic entities that are close to a given one. Another example of this type of problem is connectivity analysis in a network, which includes finding whether two geographic entities are connected in a network.

On the other hand, in the field-based conceptual view of space this problem is solved by applying analysis techniques based on the application of the operations of a map algebra among a set of function fields. Some of the problems that can be solved with this techniques are slope and aspect analysis, flow analysis, visibility analysis, and viewshed analysis (points visible from a given point under certain conditions).

A third type of problem is the task of helping in the decision-making process, which builds on the techniques to compute properties of geographic information. The general question that describes this kind of problem is *What location satisfies these requirements?* In this type of problem, the user gives the conditions of the problem that might include geographical and non-geographical requirements, and the GIS tool must find the best location that fulfills these requirements. A level of complexity is added with respect to the previous kind of problems because not only relationships between the available geographic information must be computed, but also the optimum solution for a problem must be found.

Consider for instance the problem of placing a new TV relay station. The GIS tool that solves this problem must maximize the range of the station while minimizing the cost of building it caused by new roads, or clearing forests. Another typical example of this type of problem is the process of finding a route between locations in a road network that minimizes the distance traversed or the time needed. Network analysis is one of the cornerstones of GIS functionality and applications may be found in many areas, from transportation networks to utility management.

A fourth (and final) type of problem involves making predictions regarding geographic information. This problem includes answering the question *What will be the situation in the future?* This is the most complex kind of problem because it involves defining a model of real-world that uses geographic information to simulate real-world behaviours and predict future information. The archetypical example of this kind of problem are meteorological models that take temperature, precipitation and atmospheric pressure at a given time to predict the weather conditions in the future.

2.4.3 Computational Performance

We have seen in the previous sections that GIS development tools require special data abstractions for the representation of geographic information, and special operations and analysis techniques for its manipulation. Furthermore, geographic information is voluminous and complex, and analysis techniques require much computing power. Therefore, in order to achieve computational efficiency in the manipulation of geographic information, it is necessary to define new algorithms for:

- *The implementation of operations on geographic information.* This problem has been addressed by the research field of *computational geometry*, which is a branch of algorithms dealing with computation on geometric objects. An introductory text on the field can be found in [PS85].
- *The evaluation and optimization of queries.* This problem involves defining appropriate evaluation plans for queries that include operations on geographic information.

2.4.4 Summary

We can summarize the requirements described in this section as follows:

- It is necessary to provide an exhaustive set of primitive operations on the data abstractions of the conceptual model (Section 2.4.1).
- A query language for this conceptual model must be defined. This language is used to retrieve and manipulate the data abstractions of an application schema

defined using the conceptual model. The results of queries must be represented by an appropriate language that captures all the abstractions of the conceptual model (Section 2.4.2).

- The abstractions in the conceptual model must be used to provide generic solutions for common geographic problems. There must be a precise definition of the problems to be solved, the information needed, and the techniques used to solve them (Section 2.4.2).
- Finally, new algorithms for the efficient implementation of operations on geographic information, and the evaluation and optimization of queries are required (Section 2.4.3).

2.5 Data Presentation

The visualization of geographic information is a distinctive feature of GIS applications compared with general-purpose information systems [Voi94, Voi95, PR95, dOGM99]. The user interface of a GIS application should at least allow users to effectively select and retrieve spatial data, manipulate it, and present the results of geographic analysis procedures. Whereas only data retrieval and manipulation is considered a central task for relational DBMS, GIS development tools are required to handle the visualization of analysis results as well.

We have shown in the previous sections that geographic information requires specific data types and operations for its representation and manipulation. Therefore, new visualization techniques are also required to display these data types and the results of these operations. In addition to this, geographic information has some peculiarities that have an impact on the presentation process:

- *It needs different abstractions for the visualization.* The information on traditional information systems can be visualized appropriately in a user interface using the same structures and abstraction levels that are used in the logical model, whereas geographic information is represented by complex structures that can only be visualized using different abstractions than those used to represent the information in the computer. For instance, in a general-purpose information system, integer values, string values or data in a relation can be displayed in a user interface using its textual representation. However, in a GIS a line value that is represented as a list of vertices in the logical model cannot be displayed as a list of coordinate values in the user interface. Instead, a graphical representation must be used.
- *It is multi-dimensional.* In addition to using at least two spatial coordinates to reference each location, a set of descriptive alphanumeric attributes are also

associated to each location. These attributes can be considered additional dimensions for the location. However, the value must be displayed in a two-dimensional screen. Therefore, the style properties of the graphical object (e.g., line color, line width, or fill pattern) must be used to represent the additional value dimensions.

- *It is voluminous.* The process of displaying geographic information in the form of a map requires the retrieval of a large amount of data. Moreover, these data must be displayed in a device with limited size and resolution. Hence, some visualization procedure must be designed to select the relevant information to be displayed in order to minimize the data retrieved and to avoid the saturation of the display device with information.

- *It is required at varying scales and sizes.* Appropriate detail must be presented at each combination of scale and size. Unlike textual information where there is no limitation in the space used, the amount of information displayed about geographic objects must fit in the space occupied by these objects.

- *It is required from different perspectives.* Geographic information systems allow the integration of data from several different sources in a greater degree than traditional information systems. This integration must be performed by presenting data to the user in a multiplicity of ways. For instance, it must be possible to display the same data sets using different graphical representations, even of a different nature (e.g., displaying a data set in a map and as a table).

- *It is projected onto a flat surface.* Geographic information refers to the curved surface of the Earth, but it is displayed in flat surfaces such as paper or computer screens. Procedures are needed to adapt information from a curved space to a flat one keeping important properties of the information.

We describe these issues in more detail in this section. First, in Section 2.5.1 we introduce two concepts that form the basis of the visualization of geographic information: the map metaphor, and the cartographic generalization problem. Then, we enumerate some of the visualization and interaction procedures that are desirable in a user interface for geographic information in Section 2.5.2.

2.5.1 A Metaphor for Visualizing Geographic Information

In human language, a metaphor is the use of a word or phrase denoting one kind of idea or object in place of another for the purpose of suggesting a likeness between the two. In a computer system, a metaphor is the use of concepts from a domain of experience familiar to the end-user in a different domain of experience in order to facilitate its understanding. The knowledge that the user has from the first domain is used to structure the second domain, which may be new to the user. Common metaphors used in computer systems are the desktop metaphor of many operating system, or the shopping cart metaphor used in on-line shops.

A major research challenge on user interfaces for GIS applications is to find appropriate metaphors [EF88]. Given that paper maps have been being used for many centuries as a powerful and effective way to communicate geographic information, most user interfaces for GIS have been designed with paper maps and mapping operations in mind. Nevertheless, there are fundamental differences between paper maps and their digital equivalents:

- *Paper maps have a scale.* The *scale* of a map is defined as the ratio of distance on the map to distance on the surface of the Earth. On the other hand, the data stored in a GIS applications does not have a specific scale. Instead, it has a *resolution* that is defined as the precision in which the data was originally captured. Hence, the information in the GIS can be presented at many scales, according to the needs of the user.

- *Paper maps are static.* Once a paper map is printed, it is impossible to change the contents or interact with the information. On the other hand, a GIS application provides a user interface that enables a user to interact with the information displayed to change the view point of the map, or to request more information of any geographic entity. Furthermore, a GIS application can update the information displayed when changes occur, and represent the evolution of the geographic information by means of animations.

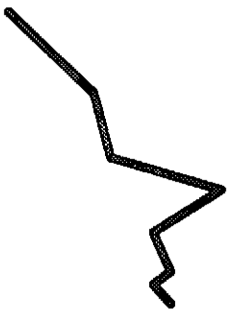
- *Paper maps are flat.* Paper maps always show two-dimensional views of the information. GIS applications can represent and display multi-dimensional information. For instance, a geographic attribute of a feature may be displayed using a two-dimensional graphical object, but a GIS application may display all the other descriptive attributes when the end-user clicks with the mouse over the graphical object.

Therefore, while the paper map is a useful metaphor for a GIS, we must be careful not to let it limit the possibilities of geographic information visualization. For these reasons, some authors prefer to consider the user interface of a GIS as a map-like view rather than as a map [Kuh91]. By using the metaphor *the-user-interface-display-is-a-map-like view*, the user interface of a GIS displays a particular rendering of the geographic information in the system according to the particular characteristics of the map requested.

Research in cognitive science has established that humans perceive, conceptualize and deal with the world at multiple levels of detail. Particularly, the user interface of a GIS must display different views of the information stored in the system according to the display parameters (e.g., map scale, display resolution), and to the demands of the user. This implies that geographic information is represented by two different abstractions at the user interface level and at the information management level. We have already defined in Section 2.3.2 a *geographic feature* as the abstraction for real world geographic phenomena that is represented and stored in the GIS. Additionally, we define a *cartographic object* as the abstraction used to visualize geographic information in a GIS application. A cartographic object is the graphical representation of a geographic feature that has been processed to be displayed taking into account properties such as map scale and display style. The process of building a cartographic object from a geographic feature is usually called *styling*.

```
LINESTRING
          (
          (0.0, 37.5),
          (11.3, 27.1),
          (13.5, 17.0),
          (28,0, 15.6),
          (19.8, 9.5),
          (21.8, 4),
          (17.5, 2.9),
          (22.3, 0.0)
          )
```

(a) Geographic Feature.



(b) Cartographic Object.

Figure 2.16: A Geographic Feature and a Cartographic Object.

To illustrate the difference between a geographic feature and a cartographic object, consider the different representations for a road section. As a geographic feature, a road section is represented using a line consisting of a list of vertices whose coordinates reference locations in the geographic space. On the other hand, a road section is represented as a cartographic object using a line whose coordinates reference positions on the screen, and with a style that determines the attributes of the graphical object. Figure 2.16 illustrates this difference. Figure 2.16(a) displays the list of coordinates of a geographic feature, whereas Figure 2.16(b) shows the cartographic object associated to that geographic feature. Similarly, the cartographic object used to represent a coverage of the field-based model is often an isopleth map composed by set of isolines, which are the locus of all points in the coverage with the same attribute value.

The mapping applied between a geographic feature and a cartographic object is important because it is used to communicate the semantics of the data to the end-user by means of the graphical rendering. Similarities and differences of graphical style between cartographic objects are used to convey similarities and differences between the respective geographic features. Some of the variables that we have at our disposal are:

- *Colors and strokes of lines and areas.*

- *Shapes and sizes of symbols.*
- *Classes and colors of coverages.*

Furthermore, not only the user interface for a GIS has to use different abstractions for the manipulation and visualization of geographic information, but also the type of graphical object used to describe a phenomenon at one scale of resolution is likely to be quite different from that at another. This implies that a geographic feature is displayed using different cartographic objects in different situations. This is referred to as the *cartographic generalization* problem in the literature [RSV93, RS94, RS95].

Consider, for instance, the user interface of a car navigation system which comprises information about roads and streets, intersections, and cities. According to the scale of the map that is being shown in the display device, different representations for the geographic information are used. In a small scale map, roads are displayed as lines, road intersections and cities are displayed as points, and streets are completely unnecessary. Moreover, the graphical object used for the cartographic objects must be computed as a *simplification* of the geographic feature because some of its details may not be visible due to technical limitations in the display device. Given that the length of a pixel in the screen may represent thousands of meters in the real-world, details of the geographic features (e.g., road bends) will not be visible because they will be represented as a single pixel.

Additional details must be displayed in the map when the resolution is increased using a medium scale map. Different graphic representations must be used for roads according to the number of lanes, or whether the road is a toll motorway or not. Cities must be represent as surfaces to indicate its exact position and extent in order to know whether a road route avoids the city or it traverses the city center and therefore traffic jams are possible. Moreover, the map must show which turns are prohibited at intersections, and which directions are possible in the entrance to a motorway. Finally, in a large scale map, roads and intersections are better represented as surfaces instead of as lines and points. Furthermore, some information like the city streets are very important while driving inside the city, but are absolutely irrelevant in other cases. Therefore, this information is displayed or not according to the concrete case.

A final issue that must be considered by the visualization metaphor is the management of information density. This issue refers to the amount of information that can be visualized simultaneously on the display device. This problem is more complex in GIS than in traditional information systems. Whereas the visualization of a large amount of information in traditional information system implies the display of many boring pages of data, the visualization of a large amount of geographic information without any management of the density of information results in a map overloaded with graphical objects that is completely unreadable. For instance, a relation with five hundred tuples may need fifty different pages of information if we consider that ten tuples can be visualized in each page. On the other hand, the amount of information

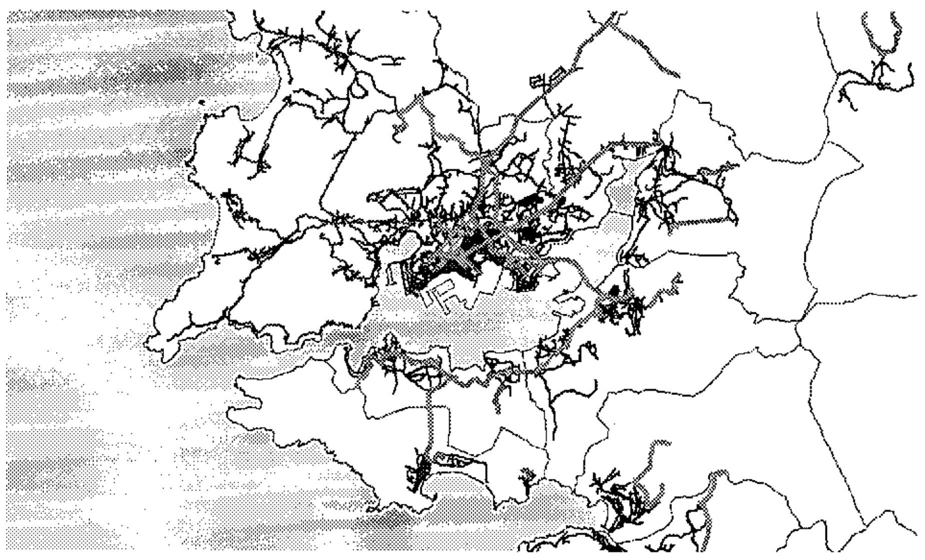


Figure 2.17: An Overload of Geographic Information in a Map.

that can be displayed on a map is limited by the physical width of the map. Figure 2.17 shows an example of this problem. The map displays the water supply network for an urban area. However, the map scale is too small to display properly the water distribution pipes. Therefore, they appear as solid color areas instead of as individual features that can be identified and manipulated.

Summarizing, by using the metaphor *the-user-interface-display-is-a-map-like-view*, a mapping between geographic features and cartographic objects must be specified considering the following issues:

- *Styling.* Given that the abstraction used for the visualization of geographic information (i.e., a cartographic object) is different from the abstraction used for its representation on a computer system (i.e., a geographic feature), it is necessary to define a mapping that determines the graphical representation of a cartographic object from the geographic feature.
- *A geographic feature may be displayed using 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 feature, because the extra detail cannot be displayed on the computer screen.
- *A geographic feature may be associated to multiple cartographic objects with different representations.* According to the visualization properties (e.g., map scale) or the purpose of the visualization, a different kind of cartographic object

may be used for a geographic feature. For instance, a road is better represented as a surface at large scale maps, and as a line at small scale maps. Similarly, 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, or as a point at a small scale map.

- *Information density in the display must be managed.* Since there is a direct mapping from the geographic space to the display device space, the amount of information that can be simultaneously displayed is limited by the display device size. Therefore, some method must be implemented to manage the information density.

It is important that the process of setting-up and managing the mapping from geographic features to cartographic objects should be hidden as much as possible from the end-user of the GIS application. For instance, a city may have multiple different representations with different resolutions at different map scales, but the end-user must not be aware of this. The desirable situation is that the only decision that the end-user should take is whether the cities must be visible or not.

2.5.2 Interaction with Geographic Information

We have introduced in the previous section some important functional requirements for the user interface of a GIS, namely:

- *The necessity of a special metaphor for the visualization of geographic information is required.*
- *The necessity of Different abstractions for the manipulation and visualization of geographic information.*
- *A geographic feature may be visualized using different cartographic objects according to the particular display properties.*

In this section, we describe other functional requirements for the user interface of GIS applications related with the interaction mechanisms that the user needs, such as composing maps using set of layers, common operations on maps (e.g., zoom, pan, measurements), or displaying context information to understand the maps.

Displaying Geographic Information Using Maps and Layers

A special characteristic of geographic information is that all data sets are related by the fact that they refer to a set of geographic locations, in addition to any other relationship they may have. This enables a new visual analysis technique that is often called *layer overlay*.

A *layer* is a set of cartographic objects that results from the arbitrary grouping of geographic features defined by the developer or the end-user. Different layers originating from different data sets that are located at the same geographic space can be displayed together in a *stack* of layers. This process is known as *overlaying layers*. When the map display is drawn, the layers are drawn from the bottom to the top of the stack. Therefore, data from the top layers is drawn over data of the bottom, which are hidden in the process. By using the metaphor *the user-interface-display-is-a-map-like-view* together with the metaphor *information-in-a-map-is-a-stack-of-layers*, an end-user can easily and efficiently integrate and analyze data from different sources. These display metaphors are a basic component in every user interface for GIS.

Common end-user operations in the *information in a map is a stack of layers* metaphor are:

- *Add a layer.* This operation enables a user to create a layer from a set of geographic features by applying a graphical style. Then, the user can add the layer to the stack at any particular position.
- *Change the stack order.* The ordering of the stack can be changed by the user. This implies that the information is drawn in a different order and data that may have been hidden before are now displayed.
- *Remove layer.* The user can remove any layer at any time.

Common Operations on Maps

In addition to the operations that manipulate the visible layers on a map, the metaphor *the-user-interface-display-is-a-map-like-view* must provide some other operations that enable the end-user to manipulate the map properties and the geographic information that is being displayed. Such operations include the following ones:

- *Change the map center.* Usually, the geographic information displayed on a map does not fit on a single screen. Therefore, only a limited area of the information is displayed at a time. This operation enables a user to change the center of the area that is being displayed. An alternative name for this operation is *pan*.
- *Change the map scale.* This operation is also called *zoom*, and allows the user to enlarge or reduce the map scale.
- *Enable the invocation of geographic analysis techniques.* User interface mechanisms must be provided to invoke the analysis techniques described in Section 2.4.2. For instance, the question *What is it at this location?* requires

a mechanism to specify a particular location at the map (e.g., a mouse click on the map), and the a mechanism to display the answer (e.g., a separate window that displays the descriptive attributes of the geographic features associated to the cartographic objects that contain the point given by the user).

Understanding Maps: Context Information

Context information is important for all information systems because it provides the end-user with additional information to understand the data that is being displayed. Nevertheless, context information is indispensable in GIS applications for the following reasons:

- *There is a transformation from geographic features into cartographic objects.* The meaning of each cartographic object will not be obvious to the viewer, and therefore this information must also be displayed in a *graphical legend* (a term that comes from cartography).
- *The map properties must be explicit.* In order to better understand the geographic information that is displayed by a map, some of its inherent properties must explicitly be displayed. For instance, the *map scale* must be visible and can be displayed both as a numerical fraction and as a bar that relates lengths in the map to lengths in the real-world. Similarly, the orientation of the map must be visible and it can be displayed with an arrow pointing to the direction of the north in the map. Furthermore, the position of the map portion that is currently being displayed must be made explicit, for instance with an overview map that shows a larger area of the geographic space with less detail, and an indication of the current viewport displayed on the map.
- *The geographic features have descriptive attributes that can be used as graphical labels for the cartographic objects.* The automatic generation of labels for cartographic objects is an open research problem. Given that the space in the map for labels is limited, the major challenge is finding a method to discriminate which labels are relevant from those that can be discarded. The alternative that is followed by many GIS tools displays the label for a cartographic object only when an action is performed by the end-user (e.g., when the mouse is moved over the object).
- *Query results are meaningless when displayed by themselves.* In order to understand the results of a query, some additional information that is not explicitly retrieved by the query is usually needed to help the user identify the results [Ege90]. For instance, the result to the query *Which are the ten gas stations closer to my current location?* is a collection of then points. If these points are not displayed over the street network and labeled with the address of the gas station (the context information), the query results cannot be understood.

Advanced Manipulation of Cartographic Objects

There are manipulation operations of cartographic objects that require advanced interaction mechanisms, which must be provided by the user interface.

Many of the geographic analysis techniques require the end-user to enter geographic values as parameters for the analysis. For instance, the query *Which is the country located under this point?* requires the user to enter a geographic point. Since geographic features cannot easily be given by the end-user as text, the user interface must provide tools to draw the geographic part of geographic features used as parameters for queries.

Furthermore, we described in Section 2.4.2 that there are two opposite ways of accessing the information in the system. In the first form, the user gives a location (geographic information) to the system and the descriptive components of the objects found is retrieved. In the second form, the user gives descriptive information of the objects to be retrieved and the result is the geographic component of the objects. The user interface must support both types of interaction, that is, it must be possible to retrieve the descriptive components of the cartographic objects displayed in the map, and it must be possible to locate the cartographic objects that correspond to some particular descriptive attributes.

Producing Paper Maps

Even though that we have repeatedly said that a GIS map is not a paper map, it must not be forgotten that a GIS application must be able to produce high-quality paper maps. Therefore, the GIS development tool must provide means to take into account the limitations mentioned before for paper maps. A particular common requirement is that the tool must provide a mechanism to produce *cartography series*. That is, it must be possible to partition the geographic space into equally-sized areas that are later printed in a paper map with a common format.

2.5.3 Summary

We can summarize the requirements described in this section as follows:

- The data abstractions used for the storage and processing of geographic information are not suitable for its presentation to the user. Therefore, new data abstractions are needed that support displaying different views of a single geographic object at different resolutions or with different visual styles according to display parameters such as scale (Section 2.5).

- There is a need for appropriate metaphors to manipulate geographic information at the user interface of the system. These metaphors must be based on the well-known map metaphor, and must incorporate dynamic operations such as *zoom*, *pan* and the addition and removal of information grouped in layers (Section 2.5.1).

2.6 System Architecture

We have described in the previous sections the functionality that is required by GIS applications, and we have given a classification of this functionality according to its type. It has been shown in these sections that the representation, manipulation and visualization of geographic information has special characteristics that make GIS different from traditional information systems. In this section, we present other characteristics of geographic information that influence the architecture of GIS applications. First, in Section 2.6.1, we describe different architectural approaches for the management of geographic information. Then, we describe in Section 2.6.2 different ways in which the functionality for geographic analysis can be implemented in a computer system.

2.6.1 Architecture Types for Data Management

Given that geographic data sets tend to be large, one of the more complex task of GIS is to efficiently manage huge databases of complex information. Traditional information systems use DBMS to manage the information, but in the past these systems did not provide support for the efficient representation and manipulation of geographic information. Therefore, it was necessary to provide alternatives to solve this problem.

The first generation of GIS development tools used a *hybrid* architecture for the management of geographic information (also called *dual* architecture). This kind of architecture consists of two independent software modules to manage the data stored in the system, one for traditional business data (i.e., a DBMS) and another for geographic information. Both components are integrated by a software layer provided by the GIS application.

In order to overcome the limitations of this approach, the second generation of GIS development tools used a *layered* architecture (sometimes referred to as *integrated* architecture). Instead of using two independent systems to manage geographic and traditional information, a single software module manages both types of information using a DBMS. Values of traditional data types are managed as usual, and geographic values are stored either using traditional data types or binary large objects. Finally, GIS applications are implemented using the functionality provided by this module.

These two approaches have proven inefficient and inappropriate, and nowadays the preferred solution by the third generation of GIS tools is the *extensible* architecture. In this approach, the functionality for the management of geographic information is implemented within the DBMS in order to provide efficient representation and manipulation of geographic information. Geographic data is manipulated at the same level as the rest of information in the DBMS, instead of being manipulated by a external module. Now, we describe each of the approaches in more detail.

Hybrid or Dual Architecture

The hybrid approach to GIS architecture uses two different subsystems to manage geographic data and non-geographic data (see Figure 2.18). Typically, the geographic data management subsystem uses files of the operating system to store the values using proprietary data formats and structures. On the other hand, non-geographic data are managed by a DBMS. Pointers from records in files to identifiers of tuples in the database and vice versa are used to link elements in both subsystems.

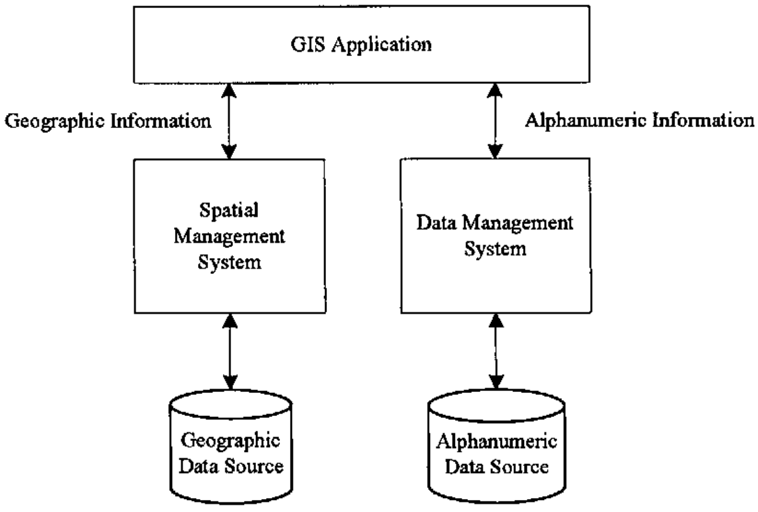


Figure 2.18: Hybrid Architecture.

The motivation behind the hybrid architecture for GIS is two-folded:

- Management strategies for geographic data are very different from the ones for traditional data, and therefore a different system is needed.
- Existing software for the management of spatial information such as CAD systems can be used as the geographic information management subsystem. This

allows to reuse easily existing software modules and data.

However, this architecture suffers from two major disadvantages. First, two heterogeneous data models coexist, which implies difficulties in modeling, querying, optimization, and integration. As an example, a query that combines geographic and non-geographic operations must be split in a query for each individual subsytems, and the results must be processed and integrated by the GIS application. And second, the geographic data values are stored outside the database and therefore cannot benefit from standard database functionality such as concurrent access management, integrity, security, reliability, or recovery techniques.

Layered Architecture

The layered approach to GIS architecture appears to overcome the limitations of the hybrid architecture by storing geographic and non-geographic data using exclusively functionality provided by a DBMS (see Figure 2.19). Geographic analysis operations are implemented in a software module that uses standard relational DBMS techniques for accessing the stored data.

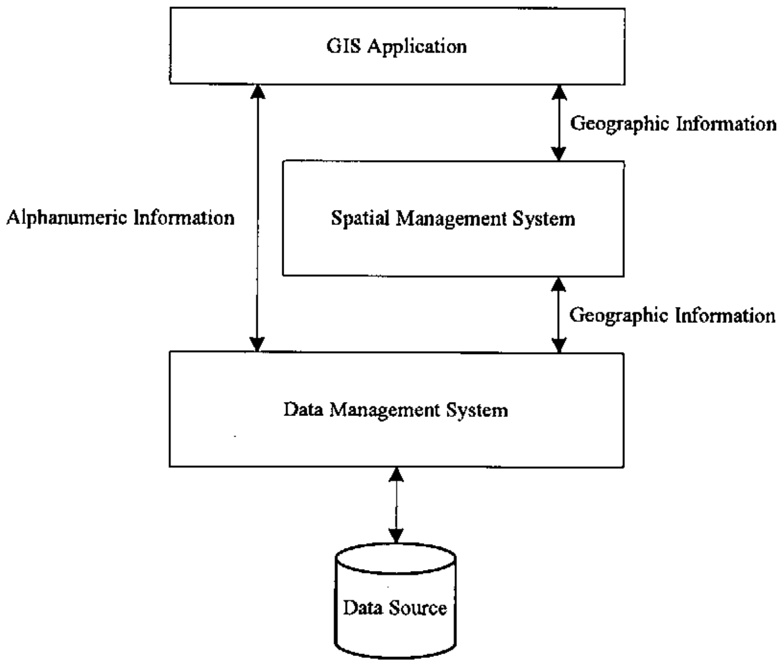


Figure 2.19: Layered Architecture.

There are two alternatives for this, according to the way in which geographic data values are stored in the DBMS:

- *Pure relational.* Geographic information is stored in relations and traditional relational data types. For instance, a polygon value is represented a set of tuples, each representing a vertex of the polygon using integer or float values for the coordinates. Every GIS operation is implemented using relational operations provided by the DBMS.

- *Opaque relational.* Geographic data values are stored in the binary large object abstraction (*blob*) provided by the DBMS. Every GIS operation is implemented by the spatial management system outside the scope of the DBMS.

These approaches suffer from drawbacks that make them inappropriate. Briefly stated, the implementation of geographic operators and display procedures is complex and inefficient. More particularly:

- *It violates the data independence principle.* In the pure relational approach, formulating queries on geographic information requires a knowledge of the structure of the geographic information. In the opaque relational approach, the format of the geographic information in the large objects is often not public and the GIS development tool cannot be switched. In both approaches, changing the structure implies a deep reorganization and changes in the query formulation.

- *Performance on retrieval of spatial data is poor.* The pure relational approach requires a considerable amount of relational tuples to represent a geographic data value. Therefore, a large number of relational accesses and joins are necessary to reconstruct the geographic value and to connect it to the other attributes of the geographic feature. In the opaque relational approach, the DBMS is completely ignorant of the structure of geographic information. Therefore, query algorithms (such as spatial join) must be implemented outside the DBMS without any query optimization (e.g., a spatial join must be implemented loading using a memory-based join algorithm). Furthermore, indexing mechanisms for geographic information cannot be implemented, and therefore, query optimization is not possible.

- *It lacks user friendliness.* The end-user has to manipulate tables of points in the pure relational approach, and large objects in the opaque relational approach.

- *It is impossible to express some geometric computations.* Regarding the pure relational approach, some operations (e.g., adjacency test, point query, or window query) cannot be expressed with SQL. In the case of the opaque relational approach, all operations must be implemented outside the scope of the DBMS and must be expressed in the language defined by the spatial management system instead of the query language of the DBMS.

The major advantage of these approaches is that they can be implemented on top of many existent relational DBMS. Moreover, the pure relation approach benefits from using only standards for the representation of the information (SQL and the relational data model). On the other hand, since the opaque relational approach requires little functionality from the underlying management system, it can be implemented on different systems beyond relational DBMS (e.g., file systems, legacy systems, object-oriented DBMS).

Extensible Architecture

The solution that is recently starting to be implemented by many DBMS and GIS development tools to manage geographic information is the *extensible approach*. DBMS extensibility has been receiving a growing interest in the past years in many new application domains. Database management systems have evolved from the relational data model to more advanced data models that can be extended by developers or final users with the definition of new data types, data models, operations, query languages, or optimization procedures. Extensible DBMS [GDF$^+$99, DG00a, Die01] are no longer research prototypes. Commercial systems such Informix Universal Server, Oracle or PostgreSQL can be extended by means of user-defined types and procedures, which can be used later seamlessly in the query language.

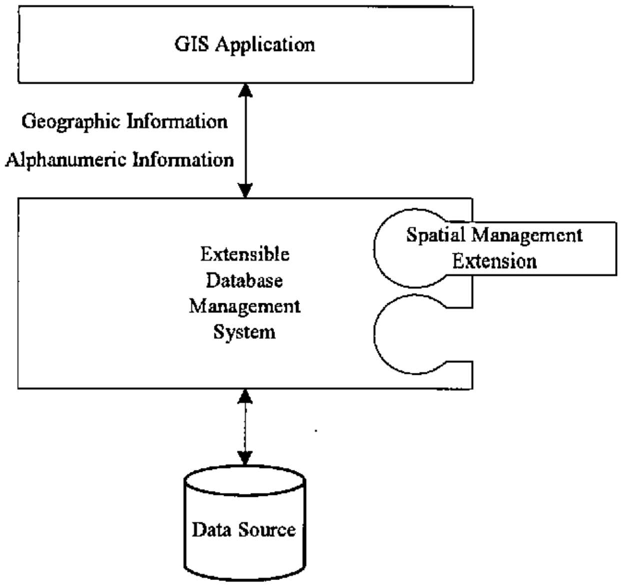


Figure 2.20: Extensible Architecture.

In the case of GIS, an *extension module* is provided for the DBMS that extends the data model with new data types and operations to represent and manipulate geographic information, together with access structures and optimization methods that take into account the new data types [Dav98, Adl01] (see Figure 2.20).

Query optimization can be performed in this approach, as opposed to the previous two approaches, because the structure of geographic information is no longer opaque to the DBMS. Furthermore, queries are more user-friendly because the geographic operations are integrated with the query language.

2.6.2 Architecture of the System

The separation of functionality described in the previous section can be achieved by many different architectures, ranging from a monolithic architecture running in a single computer to a completely distributed, service-based architecture. Each different architecture has advantages and drawbacks that must be considered, and the selection of one of them must be based on the implications that each approach has on the functionality.

A first classification of the different architectures for information systems distinguishes *monolithic architectures* from *client/server architectures*. A monolithic architecture implements the functionality in such a way that it cannot be separated in different functional components. This approach is no longer used in information systems, and therefore we will concentrate only in client/server architectures. In this kind of architecture functionality is broken down into server-side and client-side tasks. Server-side tasks are performed by a server computer that implements a service providing the functionality of the task. A client computer performs the client-side tasks and delegates server-side tasks on a server computer that is accessed using a network.

Different types of client/server architectures can be implemented, according to the amount of functionality that is implemented at the client-side. In a *thin-client system* as few functionality as possible is implemented as client-side tasks. On the other hand, a *thick-client system* implements as much functionality as possible as client-side tasks.

Furthermore, client/server architectures can also be categorized according to the degree of distribution of server-side tasks. In a *distributed system*, the server-side functionality is broken down into many services that interoperate to achieve a more complex task. In *traditional client/server architectures*, all server-side functionality is implemented in a single service.

Now, we describe these issues in more detail, and we analyze the advantages and drawbacks of each approach.

Thick and Thin-client Architectures

In a thin-client system, the functionality of the client is limited to display the user interfaces and all functionality of the application must be implemented by the server. On the other hand, a thick-client implements much of the functionality of the system, such as some data processing functionality. The advantages and drawbacks of these approaches are complementary. Therefore, we will enumerate them simultaneously.

- *Communication needs.* In a thin-client, each functionality requires communication with the server, whereas in a thick-client, some functionality can be performed at the client. On the other hand, data transmitted from the server to the client in a thin-client architecture are completely processed, whereas in a thick-client architecture the data are still unprocessed, and therefore of a larger size.

  As an example, in a thin-client, *pan* and *zoom* operations on the map require a request to the server to compute the new view of the map, which is then transferred to the client. There are many data transmission operations, each sending a small portion of the map to the client. On the other hand, in a thick-client, the server sends a big portion of the map data to the client in a raw format that is afterwards rendered by the client. In this case, the pan and zoom operations do not require new data transmissions.

  This implies that in a thin-client architecture, the fundamental part of the communication is caused by multiple processing requests, whereas in a thick-client architecture, it is caused by transmitting large amounts of unprocessed data.

- *Functionality updates.* In a thin-client, the control over the functionality is at the server side, whereas in a thick-client, some responsibility is delegated to the clients. This implies that updates in the functionality are easier in a thin-client architecture because little change is required to the clients. On the other hand, a change in functionality in a thick-client architecture may imply an update of all installed clients, or the need to maintain backward compatibility with old clients.

- *Technological requirements.* The power required from a platform to run the client of a thin-client architecture is limited to displaying the user interface and communicating to the server. On the other hand, much more power is required for a platform to run the client of a thick-client architecture (e.g., support for a programming language). For instance, a web-based thin-client can be implemented using only HTML and small scripts in JavaScript. On the other hand, a thick-client may require a Java applet or an ActiveX control to be embedded in the web application.

- *Functionality restrictions.* Given that all the functionality of a thin-client architecture is implemented at the server-side, it is not difficult to implement interaction mechanisms more complex than single-click operations. This is not the case in a thick-client architecture.

On the other hand, the client component in a thick-client architecture must be developed specifically for each platform, since interaction mechanisms vary from platform to platform. For instance, a thick-client developed using the ActiveX technology cannot be used in a UNIX machine. Similarly, the interaction mechanism using a mouse in a Macintosh computer, which uses a single-button mouse, are different from those in a PC, which a multiple-button mouse.

GIS applications must find a compromise between these two architecture approaches, taking into consideration the particular characteristics and limitations of the runtime environment.

Distributed Architectures

Recent developments in information technology have resulted in a number of distributed object architectures that provide the framework required for building distributed applications (e.g., the Distributed Component Object Model architecture (DCOM) provided by the operating system *Windows*, the Java Remote Method Invocation protocol (RMI) provided by the Java language, or CORBA). These frameworks also support a large number of servers and applications running concurrently. The general idea of the distributed architecture service model for GIS is that the functionality of GIS applications is split into many independent services with simple interfaces. Complex tasks are performed by locating and chaining a collection of services.

The main advantage of a distributed architecture is that it provides a natural mechanism for interoperability because services are easily reusable. On the other hand, the drawback of this approach is that combining services to perform complex tasks adds another level of complexity to the architecture. Moreover, considering that the functionality of the system is divided into different software components that may be located at different computers, the data format used to transfer information from one component to another is of great importance.

2.7 Metadata for Geographic Information

Data providers and end-users have realized in the past years the importance of a detailed description of the data sets and data services in an information system. This kind of information is not related to the description of real-world phenomena, but rather to the characterization of the information regarding those phenomena and the services providing the information. The term used for this information is *metadata*.

Any specialized application domain for information systems needs particular metadata elements to describe the particularities of the application domain to other users. The field of GIS applications is no exception to this rule, and therefore, a formalization of the structure of metadata enables the following tasks:

- *To find geographic information data sets.* Metadata can be used as a catalogue to help search and find information on any given topic. Examples of metadata useful for this task are a geographic and temporal reference frame, or a description of the data entities.
- *To check whether the information satisfies some requirements.* Catalogue data for geographic information can describe the resolution, the accuracy and other information that helps the user determine whether the data set can be used according to the requirements of the application.
- *To describe how the information can be used.* Finally, metadata can describe important information that is needed to use the data set in a GIS (e.g., the geographic reference system of the data, or the storage format).

To show the importance of metadata for geographic information, it is only necessary to see the high number of initiatives (both government- and industry-promoted) to provide a standard framework for geographic information metadata [ISO03e, OGC01b].

2.8 Customization for GIS

We have presented in Section 2.4 a number of operations for the manipulation of geographic information, and a categorization of geographic problem-solving techniques. However, the amount of possible analysis techniques for geographic information is unlimited. Therefore, no GIS development tool can provide all possible operations and problem-solving techniques. As a consequence, the architecture of a GIS development tool must be *modular*, *flexible* and *extensible* to support the addition of new sophisticated data analysis procedures. The process of extending the functionality of a GIS development tool is usually referred to as *customization*.

Customization can be as simple as allowing a developer to modify the user interface of the tool by removing or changing menu options or buttons, or as complex as adding new modules that implement new analysis tasks. In the following list, we enumerate some of the features of a GIS development tool that are of great importance for its customization.

- *The GIS architecture.* In order to allow customization, the interfaces to the modules of the architecture of the GIS development tool must be exposed to developers. The power of the exposed interfaces and their simplicity are two conflicting features for which a trade-off must be found.
- *The extension mechanism.* The types of computer languages supported by the GIS development tool to develop extensions is a very important characteristic for customization. Particularly, supporting standard development languages (e.g., Java, Visual Basic, or Visual C++) or offering extension interfaces accessible via standard technologies (e.g., CORBA, or OLE/COM), are preferred over proprietary languages and extension interfaces.
- *The developing environment.* The development tools offered to facilitate the customization are of great importance. The presence of *wizards* and *code generators* facilitate the task of the developer.
- *The documentation.* A precise description of the interfaces and the functionality of the modules are very important for the developer of an extension module.

2.9 Temporal GIS and Spatio-temporal Databases

The temporal component of geographic information was systematically ignored in the previous sections. This has also been the case in GIS development tools and the more general research field of spatial databases. However, many research advances have been made in the field of temporal database during these years [DDL03]. Similarly, research and commercial tools for temporal databases have ignored the special case of geographic information changing over time.

The project CHOROCHRONOS[1], initiated by the European Commission in August 1996, aimed to achieve systematic interaction and synergy between these two areas [KSF$^+$03]. The project co-ordinator was Prof. Timos Sellis. The nodes that collaborated in this project are shown in the following list, together with the supervisor at each node.

- National Techical University of Athens (NTUA) Computer Science Division, Greece. Prof. Timos Sellis
- Aalborg University, Department of Computer Science, Denmark. Prof. Christian Jensen
- FernUniversität Hagen, Praktische Informatik IV, Germany. Prof. Ralf Hartmut Güting

[1]I was a member of this project from August 1st, 1998 to the end of the project on July 31st, 2000. The first steps of my training as a researcher were taken during those years.

- Universita Degli Studi di L'Aquila (UNIVAQ), Dipartimento di Matematica Pura ed Applicata, Italy. Prof. Enrico Nardelli
- Univ. of Manchester Institute of Science & Technology (UMIST), Department of Computation, United Kingdom. Dr. Manolis Koubarakis and Dr. Babis Theodoulidis
- Politecnico di Milano (POLIMI), Dipartimento di Elettronica e Informazione, Italy. Prof. Barbara Pernici
- Institut National de Recherche en Informatique et en Automation (INRIA), Projet VERSO, France. Dr. Stephane Grumbach and Prof. Michel Scholl
- Aristotle University of Thessaloniki (AUT), Department of Informatics, Greece. Prof. Yannis Manolopoulos
- Agricultural University of Athens (AUA), Informatics Laboratory, Greece. Prof. Nikos Lorentzos
- Technical University of Vienna (TU VIENNA), Department of Geoinformation, Austria. Prof. Andrew Frank
- Swiss Federal Institute of Technology, Zurich (ETHZ), Institute for Information Systems, Switzerland. Prof. Hans-Jorg Schek

The main result of the project derives from the emphasis that was put on discovering those concepts that are clearly spatio-temporal and cannot be obtained by a simple aggregation of temporal and spatial dimensions. Within this project, time-changing geographic information is not considered a collection of snapshots of a set of geographic objects. Instead, the new concept of *moving object* databases has been coined by the researchers in the project to refer to a DBMS that stores the evolution of a spatial object with respect to time.

The contributions of this project to the field have been enormous in many different areas. We enumerate here some of these areas:

- *Ontology, structure, and representation of space and time.* This involved the study of temporal and spatial ontologies, including their interrelations and their utility in STDBMS [Fra03].
- *Models and languages for STDBMS.* The focus here was on three topics: (i) the study of languages for spatio-temporal relations, (ii) the development of models and query languages for spatio-temporal databases [GBE$^+$00, CFG$^+$03, Cot01, Viq03], and (iii) the provision of techniques for designing spatio-temporal databases. This work built on previous proposals and covered relational, object-oriented and constraint databases.
- *Graphical user interfaces for spatio-temporal information.* Research in this area had two goals: (i) to extend graphical interfaces for temporal and spatial

databases [Lua01], and (ii) to develop better visual interfaces for specific applications (e.g., VRML for time-evolving spaces).

- *Storage structures, indexing techniques, and query processing algorithms for spatio-temporal databases.* Techniques for the efficient evaluation of queries were the focus of this area. These techniques included high-level algebraic optimizations, new indexing techniques [TVM00], and low-level strategies for page/object management [DGL00].
- *Architectures for STDBMS.* The study of alternative architectures and implementation techniques for STDBMS was of high interest. As a result, several prototype STDBMS were developed in the project to demonstrate the innovations of more theoretical work.
- *Applications of STDBMS.* Finally, applying STDBMS in realistic problem settings guided the research throughout the project, not only to traditional target applications (e.g., GIS), but also to more advanced ones where the connection to a standard database approach was not obvious.

2.10 Summary

In this chapter, we have reviewed the special characteristics of GIS applications regarding the representation, manipulation, and visualization of geographic information. We have also presented a state of the art of the models and techniques used in the process of developing GIS applications [BCLV01, BCLV03].

However, part of the analysis performed is completely new and it is a part of the work presented in this Thesis. Particularly, the design of a large object extension that automatically clusters trees of nested large objects in the physical model of a GIS application [DGL00].

Chapter 3

International Standards for Geographic Information Systems

3.1 Introduction

The proposal of a generic architecture for geographic information systems must be based on:

- An exhaustive analysis of the special characteristics of geographic information with respect to its representation, manipulation and visualization in a computer system. This analysis was performed in Chapter 2.

- A thorough review of the existing standards for GIS applications, which is presented in this chapter.

In this chapter, we review the standards published by the OpenGIS Consortium and the ISO Technical Committee 211 pointing out their strengths and their drawbacks. The rest of this chapter is structured as follows. First, we briefly describe the background and motivation of these organizations in Section 3.2. In Section 3.3, we describe the architecture proposal of the OGC and the ISO. Then, Section 3.4 describes the standards and specifications for the representation of geographic information, which includes much of the work carried out by these organizations. In Section 3.5, we describe the specifications developed to enable the implementation of geographic data services. Section 3.6 is devoted to the description of services for the portrayal of geographic information. Then, we analyze brieflythe OGC and ISO proposals in Section 3.7, where we present the components that are still missing from the architecture proposal and the drawbacks and missing functionality that can be found in the existing components. Finally, we summarize the contents of the chapter in Section 3.8.

3.2 Background and Motivation

Geographic information can be found at the foundation of many engineering, research and management tasks (e.g. architecture, civil engineering, environment management). Given that the requirements of each application domain depend heavily on the particular tasks that are supported by the applications on that domain, many different types of computer applications have been developed focused on many different aspects of geographic information. For instance, civil engineering requires precise representation of information that is limited to restricted geographic areas (e.g., an exact representation of a river valley to build a bridge across it). In this case, tools for measuring distances, areas and volumes are more important than different map projections algorithms. On the other hand, environment management applications require the representation of vast areas with information acquired from satellite images (e.g., a representation of a complete river basin to analyze the impact of an industrial spill). The tools that are more important in this chase are those that can be used to interpret the information acquired by the satellite, to manipulate the data, and to derive new information.

Given that each application has a different point of view on geographic information, each developer has defined conceptual models, geographic data models, storage formats, analysis operations, or representation procedures specially adapted to the requirements of the application. As a consequence, there is nowadays a problem of interoperability between the tools: it is not easy to use one GIS development tool to analyze the information collected with another tool.

In order to solve this problem, a number of government, research and industry partners founded the OpenGIS Consortium (OGC) in 1994 to promote interoperability among GIS development tools. The mission of the OGC is to define and divulge standards to represent and manipulate geographic information. This mission is achieved through organizing interoperability projects, working toward consensus, formalizing OGC specifications, developing strategic business opportunities and standards partnerships, and promoting demand for interoperable products [OGC03a].

Another standards organization that has devoted much effort to GIS applications is the International Organization for Standardization (ISO) by means of the ISO Technical Committee 211 (ISO/TC 211), named *Geographic Information/Geomatics*. ISO/TC

211 is currently preparing a family of geographic information standards in cooperation with other ISO technical committees working on related standards.

The purpose of both organizations is to create specifications of standards concerning geographic information with detail enough to enable developers to create implementations conforming to these standards that interoperate without problems. Given the vast extent of this goal, the specifications are developed by a set of cooperating special interest groups and working groups, and are later adopted by a process of consensus among its membership. Both organizations break down the work of developing these specifications into smaller units in order to assist parallel developments. These units are called *abstract specification topics* by the OGC, and simply *international standards* by the ISO. In order to illustrate the kind of work that is being carried out by these groups, we have listed in Appendix A the abstract specification topics proposed by the OGC, the international standards published by the ISO/TC 211, and the international standards that are still in a draft state.

For many years, the OGC and the ISO were working independently to reach overlapping goals, but nowadays both bodies seek to converge toward a common solution. This implies that there is no competition between the OGC and the ISO in the process of defining abstract fundamental specifications. Nowadays, it is accepted that the ISO/TC 211 must deal with long-term, abstract, static standards, and the OGC must work on industry-oriented, technology-dependent, evolving standards. As a consequence, the first versions of the OGC abstract specification topics were developed independently by the OGC working groups and where different by the proposals of the ISO/TC 211. Currently, the OGC has decided to replace the relevant topics in the abstract specification by the international standards adopted by the ISO/TC 211.

When individual topics of the abstract specification are sufficiently mature, the OGC proposes *implementation specifications*. These specifications are platform-neutral adaptations of the abstract specifications that take into account the limitations of particular computer technologies, or describe a simple subset of the abstract specification to enable easy and fast software developments. We enumerate some of the implementation specifications proposed by the OGC in Appendix A.

Finally, in order to provide an insight into this complex set of specifications for conceptual models, services, exchange languages and many other components of the architecture of a GIS, both organizations also publish a reference model [OGC03a, ISO02a].

3.3 The OGC and ISO Target Architecture

The final goal of the work being carried out by the OpenGIS Consortium and the ISO/TC 211 is the complete definition of a generic architecture for GIS applications. This is done by specifying standards for the components that build this architecture.

Even though this work is far from complete, a general idea of the architecture can already be seen.

First, we can classify the standards proposed by these organization according to their nature in the following categories:

- *Models.* These standards define a model for characterizing and encoding geographic information and related support information. They can be *general models* for geographic information (e.g., the *general feature model* [ISO02e], the *spatial data model* [ISO03b] or the *service model* [OGC02b]), or *application domain models* that characterize application schemas and business processes for specific domains.
- *Languages.* Some specifications define languages to encode the semantics, syntax and schema of geographic-related resources. An example is the geography markup language specification (GML [OCG03a]), which is an XML application for the representation and exchange of geographic data values in a platform-independent way.
- *Services.* A service definition models a business process by encapsulating distinct parts of the functionality present in the business process. A service is specified by giving a formal definition of its interface as a set of operations with names, parameter lists, return values, and processing actions. An example of this type of specifications is the Web Map Service specification [OCG02c, OCG02a] that defines a web-based service for the portrayal of geographic information.

Even though the work of the OGC and the ISO/TC 211 is far from complete, the models, services or languages already defined in the specifications can be organized in the three-tier architecture depicted in Figure 3.1 [OGC02b]. The tiers of the architecture are those characteristic of general-purpose information systems, but the components within each tier are specifically designed for the efficient representation, manipulation and visualization of geographic information. Each tier is in charge of the following tasks:

- *Information management tier.* It is responsible for physical data storage and data management, including datasets, conceptual schemas and metadata.
- *Processing tier.* It is charge of large-scale computations involving substantial amounts of data.
- *Human interaction tier.* This tier is responsible for physical interaction with the user through display and input media, and an appropriate dialogue.

Some of these models, services or languages do not fit neatly within one of the tier. Instead, they provide support for other components of the architecture (this is

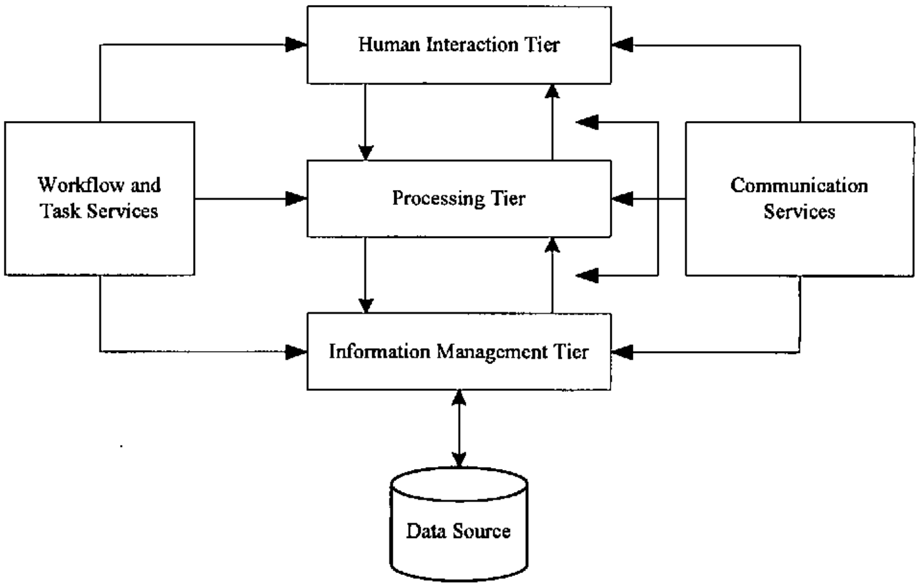


Figure 3.1: OpenGIS Logical Architecture.

represented in Figure 3.1 by the components at both sides of the figure that interact with all levels of the system). We can classify these components into these two categories:

- *Workflow and task services.* They provide support for specific tasks or work-related activities, involving a sequence of activities or steps that may be conducted by different persons. An example of this category is a service-chaining service. Even though no implementation specification has yet been defined, the OGC has already defined a model for combining services to achieve larger tasks [OGC02b]. This model includes the definition of architecture patterns to coordinate the execution of multiple independent services working on a single task. According to the location where the coordination of the task is performed, these patterns can be classified in the following categories:
  - *User-defined chaining.* The human user manages the workflow.
  - *Workflow-managed chaining.* The human user invokes a workflow management service that controls the chain while the user is aware of the individual services.
  - *Aggregate service.* The user invokes a service that carries out the chain with the user having no awareness of the individual services.
- *Communication services.* It connects the various services and tiers together, by encoding and transferring data across communications networks.

3.4 Standards for Representing and Manipulating Geographic Information

The foundation of any information system is always the conceptual, logical and physical models for the representation and manipulation of data. Much of the work of the OGC and the ISO has been centered on these specifications because it is the cornerstone of the architecture that must provide support for all applications build upon it.

In this section, we describe some of the standards proposed by the ISO and the OGC for the representation and manipulation of geographic information. First, we describe in Section 3.4.1 the standard specification proposed by the ISO/TC 211 of a conceptual model for geographic information. Then, in Section 3.4.2 we present the standard specification by the OGC of a logical model that profiles the aforementioned conceptual model to enable fast and simple implementations by developers. Section 3.4.3 describes the geography markup language, which is used as an exchange language for geographic information in many of the specifications. Finally, we present the standards for metadata and feature cataloguing that enable users to understand the structure and nature of the information, and allows computer systems to automatically manage and discover geographic information data sets.

3.4.1 Conceptual Model for Geographic Information

In order to enable different users and GIS tools to share geographic information, it is necessary to define common abstractions with respect to information and manipulation operations. These abstractions form a *conceptual model for geographic information*, as was described in Section 2.3.2.

In order to define a conceptual model, a set of concepts to describe the universe of discourse in an abstract way must be provided. These concepts must be defined precisely in order to enable the communication of concepts from the universe of discourse using concepts from the conceptual model. Furthermore, the definition must also include a mapping from the concepts of the conceptual model to a formal language that provides unambiguous and consistent representation of the application schema. This process is depicted in Figure 3.2.

The ISO has published an international standard [ISO02e] to model at a conceptual level geographic information phenomena and their properties. This conceptual model enables users and developers to define an *application schema* that provides a formal description of the data structures, data contents, and operations for manipulating and processing data required by one or more applications within a particular application field. The application schema may also be used to apply automated mechanisms for data management. An application schema addresses the logical organization of

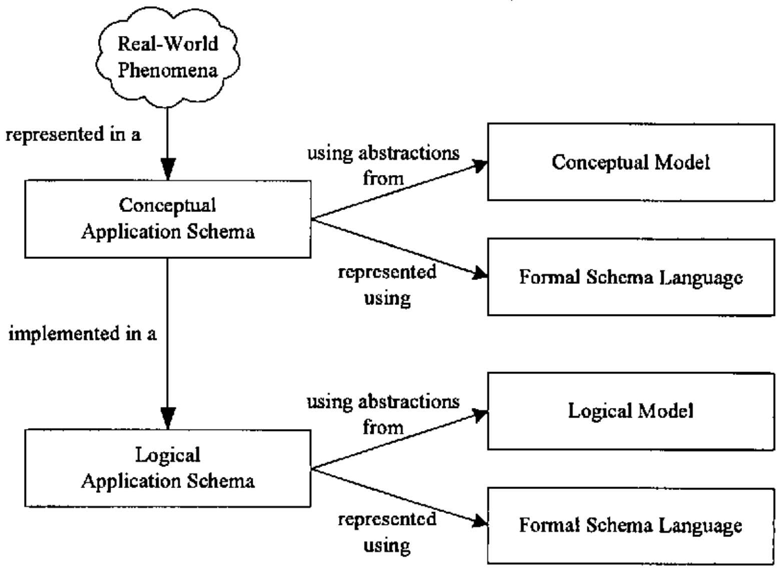


Figure 3.2: Models in an Information System.

information, rather than the physical. Such a conceptual model has not yet been adopted by the OGC, but it is probably going to be based on the ISO proposal [ISO02e] and the conceptual model implied in [OCG03a].

The conceptual model defined by the ISO/TC 211 includes the following abstractions:

- *Feature.* It is an abstraction of a real-world phenomenon and the fundamental unit of geographic information.
- *Static information.* It carries all static information of a feature, including both geographic and non-geographic properties.
- *Relationship.* It specifies the context for the feature type and its instances, and consists of a link from one feature type to the same or another feature type.
- *Behaviour.* It is represented by operations that are applied over feature instances.
- *Spatial Data Type.* It consists of an exhaustive set of data types and operations for the representation and manipulation of spatial data, such as *GM_Point*, *GM_LineString*, or *GM_Polygon*. These abstractions are defined in a different ISO International Standard [ISO03b].

In this section, we describe these abstractions of the conceptual model in more detail. The rules for mapping these concepts into a formal language are outside the scope of our work, and therefore will not be described.

Features

The fundamental abstractions of the conceptual model is called *feature*. A feature is an abstraction of a real world phenomenon and is the basic unit of geographic information. Each phenomenon of the world that is important in the application schema is represented by a feature.

The characteristics of a real-world phenomena are carried in a feature by means of *feature properties*. These characteristics are derived from the universe of discourse as perceived in the context of an application domain, and can be from one of the following types:

- *Static information*. This is represented in the conceptual model using *feature attributes*.
- *Context information*. The conceptual model represents this information using *feature associations* between features.
- *Behaviour information*. This is represented in the conceptual model by *feature operations*.

Given that the goal of a conceptual model is to provide a set of tools to describe concrete application schemas, it is necessary to include several concepts in the conceptual model in order to describe types (or classes) of features and types (or classes) of associations among these features. A *feature type* is the class for all features that represent the same information with the same semantics. Each feature that belongs to a feature type is called a *feature instance*. Feature types are equivalent to classes and feature instances are equivalent to objects in object-oriented modeling.

Figure 3.4[1] shows an example of features that represent real-world phenomena. The real world phenomena *the country named Spain*, *the country named Portugal*, and *the country named France* are each represented by a feature instance. The attributes of each feature describe the information of the countries that is relevant for the application that is being modeled (e.g., country name, population, or government type). The context information for each feature may include the existence of extradition treaties among countries represented by an association between the feature instances that represent the countries. Finally, the behaviour information may include the representation of

[1]The information in this figure comes from the CIA World Factbook (http://www.cia.gov/cia/publications/factbook/) as of 11 May, 2004.

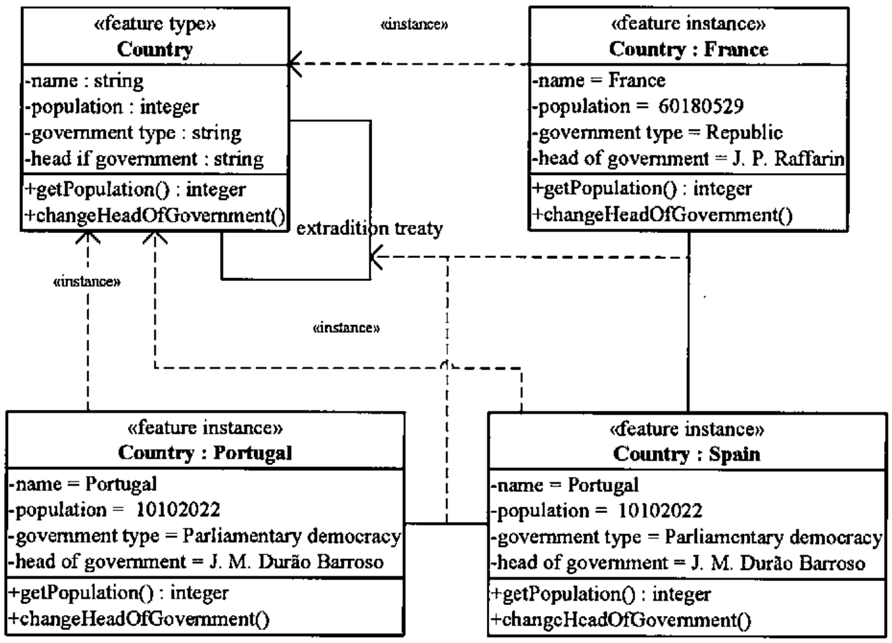

«feature type»
Country
-name : string
-population : integer
-government type : string
-head if government : string
+getPopulation() : integer
+changeHeadOfGovernment()
«instance»
«feature instance»
Country : France
-name = France
-population = 60180529
-government type = Republic
-head of government = J. P. Raffarin
+getPopulation() : integer
+changeHeadOfGovernment()
extradition treaty
«instance»
«instance»
«feature instance»
Country : Portugal
-name = Portugal
-population = 10102022
-government type = Parliamentary democracy
-head of government = J. M. Durão Barroso
+getPopulation() : integer
+changeHeadOfGovernment()
«feature instance»
Country : Spain
-name = Portugal
-population = 10102022
-government type = Parliamentary democracy
-head of government = J. M. Durão Barroso
+getPopulation() : integer
+changeHeadOfGovernment()

Figure 3.3: Feature Types Classifying Feature Instances.

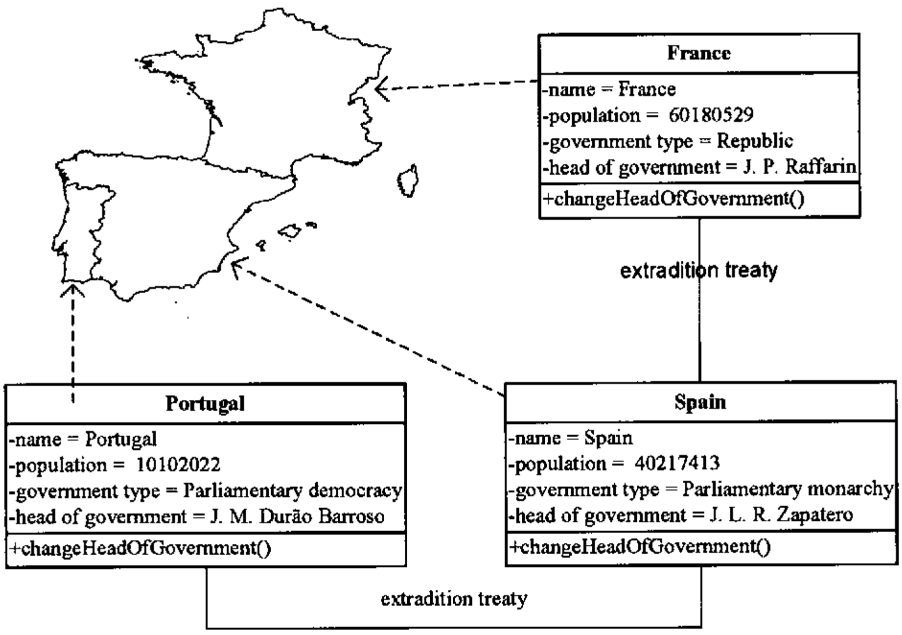

France
-name = France
-population = 60180529
-government type = Republic
-head of government = J. P. Raffarin
+changeHeadOfGovernment()
extradition treaty
Portugal
-name = Portugal
-population = 10102022
-government type = Parliamentary democracy
-head of government = J. M. Durão Barroso
+changeHeadOfGovernment()
Spain
-name = Spain
-population = 40217413
-government type = Parliamentary monarchy
-head of government = J. L. R. Zapatero
+changeHeadOfGovernment()
extradition treaty

Figure 3.4: Features Representing Real-World Phenomena.

the process of changing the prime minister of a country by means of an operation that performs this process.

Figure 3.3 shows the previous example extended to include a classification of feature instances using feature types and association types. We define a feature type named *country* that represents the countries of the world. This feature type includes the attributes, association types, and operations described previously. Some feature instances of this feature type are the features that represent the countries named Spain, Portugal and France. Similarly, some association instances of the association type *extradition treaty* are the associations between Spain and Portugal, and Spain and France.

In order to further classify feature types, the conceptual model allows users to define inheritance relationships between feature types. This is the relationship that occurs between a feature type (called the *supertype*) and one or more specialized feature types (called the *subtypes*). Any member of a specialized feature type is also a member of the general feature type, and therefore a subtype inherits all properties of its supertype. A feature type can act as supertype in more than one inheritance relationships, each having a different purpose. Additionally, the conceptual model enables the user to define *abstract supertypes*, and *unique instance supertypes*. An abstract supertype is a feature type that is not allowed to have proper instances. Instead, all its instances must be an instance of any of its subtypes. A unique instance supertype is a feature type whose instances shall not be an instance of more than one of the subtypes.

Finally, another concept defined by the conceptual model is a *constraint*. A constraint represents a rule that must be fulfilled by the feature instances and the association instances in a particular application that follows the application schema. A constraint thus prevents the creation of erroneous data and ensure the integrity of the data. Both a feature type, its properties, and its association types may have constraint definitions. An example of a constraint is a rule that asserts that the population of a country cannot be less than zero. Another example is a rule that asserts that a country cannot have an extradition treaty with itself.

As a summary, an application schema defined using this conceptual model consists of a set of feature types for the abstractions that represent the real-world phenomena of interest for a particular application. Each feature type is defined by giving: a name, a description, a set of supertypes, a set of subtypes, a set of attributes, a set of associations, a set of operations and a set of constrains. Now, we are going to describe briefly each of these modeling abstractions.

Attributes

Attributes carry all static information of a feature, including both geographic and non-geographic properties. Each attribute is characterized by a *name*, a *data type*, and a *cardinality*. The data type of the attribute defines the set of possible values that may take the attribute. The cardinality defines the number of values associated with the attribute of the feature type.

In addition to carrying the information of the feature, attributes are the connection point with other standards of the ISO 19100 series by using as data types of the attributes the ones defined in these other standards. Therefore, the attributes can be classified in the following categories:

- *Spatial attributes.* They are used to express spatial characteristics of a feature type. The data type of these attributes is one of those defined in [ISO03b]. This kind of attributes is further described below.
- *Location attributes.* These attributes are used to represent a geographic reference of a feature by means of a geographic identifier using one of the data types defined in [ISO03d], such as *GM_Point*, *GM_LineString*, *GM_Polygon*, or *TP_Edge*.
- *Temporal attributes.* They describe characteristic of a feature related to time using a data type defined in [ISO03b], such as *TM_Instant*, or *TM_Period*.
- *Quality and metadata attributes.* They carry quality and metadata information of a feature or its properties. Data types for these attributes are defined in [ISO03e]. This kind of attributes are briefly described in Section 3.4.4.
- *Thematic attributes.* This attributes represent any other descriptive characteristic of a feature. The data types used for these attributes can either be basic data types [ISO03a] or user defined data types for the application domain.

Associations

Associations are used to specify the context for a feature type and its instances. An association type consist of a link from one feature type to the same or another feature type. Individual feature instances are linked by association instances. It is important to distinguish between inheritance relationships and associations. An inheritance relationship exists only between feature types, and there is only a single feature instance that belongs to both feature types. On the other hand, an association type between two feature types exists between two feature instances that are linked by an association instance. For example, in Figure 3.5, the feature type *Motorway* inherits from the feature type *Road*, but there is a single feature instance that belongs to one of the feature types. On the other hand, in Figure 3.6, an association between the feature type *Road* and itself is be used to represent a road intersection. In this case, there is a link between the feature types (i.e., the association type) and a pair of feature instances (i.e., the association instance).

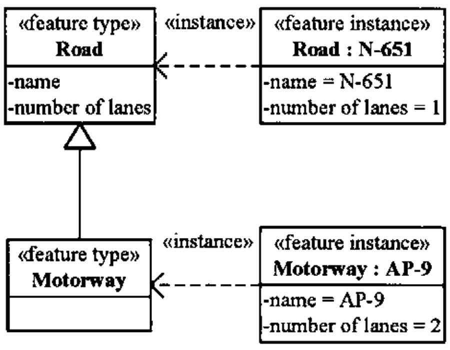


Figure 3.5: Inheritance Relationship.

An association type may have properties of its own because an association instance is in fact a feature instance. This is because the association is an abstraction of a real world phenomenon, namely the interaction between two other phenomena. This enables association types to have attributes that carry information about the association itself (e.g., spatial attributes that represent the location where the association takes place, or metadata attributes to describe the semantics of the association).

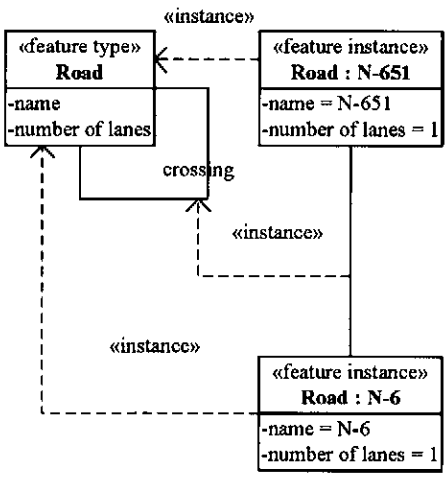


Figure 3.6: Association Relationship.

Figure 3.7 shows an example of association attributes. An association instance that represents a road intersection can have attributes that describe the directions allowed in the intersection, or a geographic point that represents the location where the roads intersect.

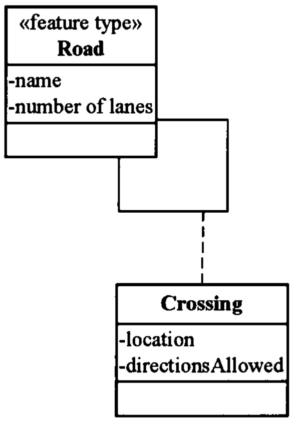


Figure 3.7: Association Attributes.

An special kind of association is called *aggregation*, and represents the relationship between features of the type *whole-part*. This kind of association is used to specify a complex feature type by describing its component parts. Moreover, there are two types of aggregation. In a *composition aggregation* (or simply *composition*) the parts live inside the aggregate and will be destroyed together with the aggregate. On the other hand, in a *shared aggregation* (or simply *aggregation*) the parts can exist even if the aggregate is destroyed.

Operations

The behaviour of feature types is represented by operations that are applied over feature instances. Operations can be classified into the following categories:

- *Observer operations.* These are operations that return the current values of attributes or computed values, without changing the state of the feature instance. An example of this category in the conceptual model described in Figure 3.3 is the operation *getPopulation()*, that returns the population value of a country.

- *Mutator operations.* These operations include actions that change the values of the feature instance. For instance, in the conceptual model described in Figure 3.3, the operation *changeHeadOfGovernment()* is an example of this category.

- *Constructor operations.* Operations of this type create an instance of the feature type for which it is defined.

Spatial Attributes

The standard that defines the conceptual model for application schemas does not provide a definition of spatial data types. These data types are defined by the ISO/TC 211 (and adopted by the OGC) in [ISO03b]. It consists of an exhaustive collection of data types for the representation of spatial data and operations on these data types for the manipulation of spatial data. These data types are defined considering a vector-based logical model for geographic information (see Section 2.3.3).

This specification defines two different hierarchies of data types considering two different points of view with respect to the structure of the underlying spatial framework: an *Euclidean space* data type hierarchy, and a *topological space* data type hierarchy. In the *Euclidean space* hierarchy, data types are defined to represent the dimension, position, size, shape, and orientation of the spatial characteristics of features. These characteristics depend on the type of coordinate reference system used to define the spatial position, and describe the quantitative properties of spatial data. On the other hand, in the *topological space* hierarchy, data types are defined to represent the characteristics of spatial data that remain invariant if the space is deformed elastically and continuously. These characteristics describe the connectivity of spatial data. Both hierarchies define an exhaustive set of data types to represent many different kinds of geometric data values. Additionally, the specification defines an algebra comprising a set of precisely-defined operations on the data types of each hierarchy to create, modify, delete and query spatial data.

Field-based Representation of Space

The spatial data types defined in [ISO03b], which were described in the previous section, only consider the object-based view of geographic space. The OGC and the ISO/TC 211 have defined a conceptual model for a field-based representation of space in [ISO02k, OGC00c]. This conceptual model represents a geographic field using the abstraction named *coverage*.

A coverage is a special kind of feature instance that associates positions in space to feature attribute values. The spatial extent of the field is represented as a set of geographic objects using the data types defined in [ISO03b]. The value of the field at each point is represented using attributes of the feature type, usually a collection of records with a common schema. Additionally, every abstraction that was defined for the conceptual model for features previously in this section can be applied to coverages (because a coverage is a special case of a feature instance, and a coverage type is a special case of a feature type).

3.4.2 Logical and Physical Model for Geographic Information

We have presented in the previous section the conceptual model proposed by the ISO/TC 211 and the OGC for the representation and manipulation of geographic information. However, the implementation of this conceptual model on a computer system is complex due to the following reasons:

- *The conceptual model for geographic values is too complex.* The conceptual model described in Section 3.4.1 uses a limited number of abstractions for the description of application schemas, and these abstractions are very similar to common abstractions in other conceptual models for general-purpose information systems (e.g., the entity-relationship model, or UML). However, the conceptual model for representing spatial data [ISO03b] is very complex because it includes two different hierarchies for the representation of spatial data, each with dozens of data types, as explained in the previous section.

- *The conceptual model is abstract.* Every conceptual model must be abstract, and therefore more details must be given in the specification of a logical model regarding the representation of the abstractions in a computer system. Moreover, computer systems are limited in their ability to represent reality (e.g., the storage space and the arithmetic precision are limited). Hence, the abstractions of the data model must be adapted to take these limitations into account.

Therefore, many different logical models are possible conforming to the standard conceptual model. In order to provide a low-cost entry point for existing players in the GIS marketplace, the ISO and the OGC have published a model for *simple feature geometry* [ISO00c] that conforms with the conceptual model for spatial data [ISO03b]. This logical model offers a simple set of data types from the Euclidean space data type hierarchy of the conceptual model, which can be easily and quickly implemented.

Furthermore, another goal of these organizations is to promote interoperability between GIS tools at many different levels. It is obviously important that the logical model used by different tools is the same in order to share information, but it is also important that the physical structures used to represent the logical model in a computer are compatible. Hence, the ISO and the OGC are developing specifications that describe precisely the way in which the logical model for simple feature geometry can be represented and manipulated in a variety of technological platforms, namely a SQL DBMS [ISO00d], a COM/OLE [OGC99a] environment, and a CORBA environment [OGC98].

These specifications are described in the rest of this section. First, we briefly describe the simple feature geometry logical model. Then, we present a brief summary of the characteristics of each particular physical model.

Simple Feature Geometry Model

Figure 3.8 shows the simple feature geometry model. The model defines a set of spatial data types that represent spatial values using two-dimensional Euclidean geometry with linear interpolation between vertices.

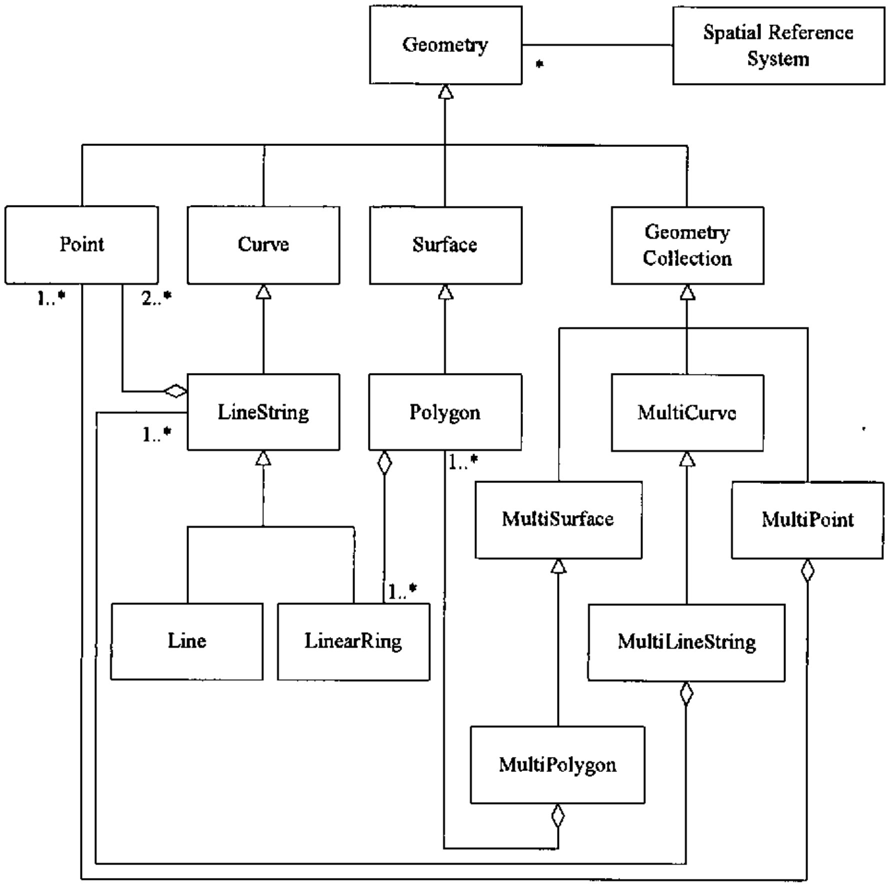


Figure 3.8: Simple Feature Geometry Model.

The root of the hierarchy is the data type *Geometry*. This data type is associated with a spatial reference system that describes the coordinate space in which the geometric object is defined. It also includes some general geometric operations (e.g., spatial predicates like *includes* or *overlaps*, or set-operations like *intersection* or *difference*). Moreover, it also includes some utility operations (e.g., converting to text and binary representations, or getting the spatial reference system).

Specific subtypes of *Geometry* are defined for 0, 1 and 2-dimensional objects, namely *Point*, *Curve* and *Surface*. The data type *Point* is instantiable and represents a point in the Euclidean plane. On the other hand, the data types *Curve* and *Surface* are abstract and they have specific instantiable subtypes for different types of 1 and 2-dimensional objects.

Subtypes of *Curve* are the data type *LineString*, which represents a set of linear segments, the data type *Line*, which represents a *LineString* with a single element, and the data type *Ring*, which represents a non-intersecting, closed *LineString*. The only subtype of *Surface* is *Polygon*, which represents a simple polygon with holes.

In addition to data types for representing simple spatial values, there are data types to represent collections of spatial values. The data type *GeometryCollection* represents heterogeneous geometry collections, and specific subtypes of this data type restrict the type of the elements in the collection, namely *Multisurface*, *MultiPoint*, *MultiCurve*, *MultiPolygon*, and *MultiLineString*.

Besides inheriting the operations defined for the data type *Geometry*, these specific subtypes define operations to access the components of the data type (e.g., get the end-points of a *Line*, or get the individual elements in a collection), and operations to compute specific properties of the data type (e.g., compute the length of a curve, or the area of a surface).

There are already some free, open source software tools that implement this specification, such as the Java-based development library *JTS Topology Suite*, the PostgreSQL extension module *PostGIS*, or the C-based development library *GEOS*.

Implementation Options

Three different implementation specifications have been published for the simple feature geometry model: a *SQL option* [ISO00d], a *COM/OLE option* [OGC99a], and a *CORBA option* [OGC98].

The SQL option stores feature collections as tables where each feature is a row in the table. Non-geographic attributes of a feature are represented using data types from the set of standard SQL data types. For geographic attributes, two different approaches are defined:

- *Based on existing SQL types.* All geographic values are represented in a separate table that contains the description of the spatial values for the geographic features. Each geographic value is represented either using a BLOB, or using a collection of rows that stores its coordinates using standard SQL numeric types (e.g., int or double). The unique identifier of the feature instance is then used as a foreign key in the geometry table to identify its geographic values.

- *Based on geometry types.* In this approach, the type system of the DBMS is extended through the definition of user-defined types for the spatial data types of the conceptual model (e.g., a user-defined type *point*, a used-defined type *line*, and so on). A geographic attribute is represented using a data type drawn from this set of user defined types.

The COM/OLE option defines a collection of interfaces to be implemented by software components in order to act as geographic data sources. Given that many data sources are not true databases, this specification does not attempt to define interfaces and manipulation mechanisms offering database functionality.

Finally, the CORBA option provides interfaces that allow GIS applications to expose functionality required to access and manipulate geographic information using CORBA technology.

The OGC provides a set of test suites to check whether a software module correctly implements the specifications. Software developers can execute these test suites and register within the OGC web site to publicize that their software tool *implements* a given specification. Moreover, software developers can submit their software tools to the OGC to be checked against the test suites by the OGC staff. Software tools that pass this test are considered compliant with the specification.

Many commercially-available software tools are registered within the OGC web site as implementing or complaint products. As an example, we can enumerate the following:

- *ESRI ArcGIS.* This GIS development tool is compliant with the OLE/COM [OGC99a] and the SQL [OGC99b] implementation specifications.
- *Oracle 9i with Spatial Option.* This DBMS is compliant with the SQL [OGC99b] implementation specification.
- *PostgreSQL with the extension module PostGIS.* This free and open-source DBMS, extended with the PostGIS module, implements the SQL [OGC99b] implementation specification.

3.4.3 Geography Markup Language

The specifications presented in the previous section enable interoperability between different GIS tools at a very low abstraction level. A GIS application can connect and use data from different data sources provided that they implement one of these standards. However, these standards are too close to the platforms for which they are defined, and cannot be easily used by an application running in a different platform. For instance, COM/OLE is a Microsoft Windows-based technology, which is very complex to use from a Unix-based application. Therefore, it is necessary to define a standard that enables data exchange at a higher abstraction level.

For this purpose, the OGC has proposed the Geography Markup Language (GML) [OCG03a], which is an XML application for the representation of geographic information. This language provides XML constructs for the representation of the abstractions proposed by the ISO conceptual model for geographic information [ISO03b] (see Section 3.4.1). That is, users and GIS applications can use GML constructs to describe features, coordinate reference systems, geographic values (both Euclidean, topological, and field-based), temporal values, or units of measure.

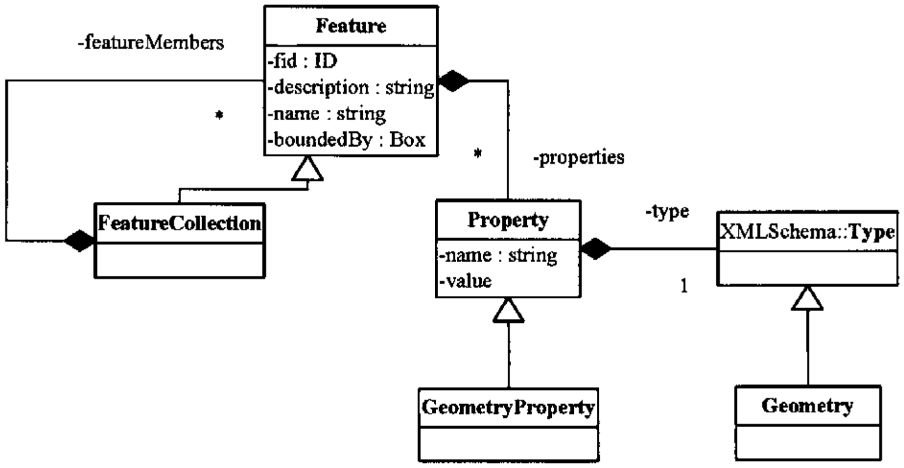


Figure 3.9: Conceptual Model For GML.

Figure 3.9 shows a simplified conceptual model of the information that can be represented using GML. A feature instance is represented by a *Feature*, which consists of a unique identifier (*fid*) a name, a description, and a bounding box. Additionally, each feature instance can be further described using a collection of *properties*. Each property consist of a name, a type defined using XMLSchema, and a value. For spatial data types, GML defines a collection of XMLSchema types using the simple feature geometry model (see Section 3.4.2, Figure 3.8). Finally, feature collections are represented by a *FeatureCollection*, which is a special case of a feature that additionally references a collection of features.

GML is a language defined using XMLSchema, and therefore enables the definition of application domain specific languages extending the abstractions defined in the specification. The previous version of the language [OGC02c] is only concerned with the representation of features and geographic values using the simple feature geometry model (see Section 3.4.2). The current version of GML [OCG03a], extends the adopted specification by offering support for the following elements of the conceptual model:

- *Exhaustive representation of geographic values.* The representation of geographic values is not restricted to simple feature geometry values. In addition to that, the language support the representation of complex geometry, non-linear geometry, three-dimensional geometry, dynamic (time-changing) geometry, and field-based information.

- *Exhaustive representation of reference systems.* In addition to spatial reference systems, the language also supports temporal reference systems, unit of measure and standards information.

- *Representation of visualization styles.* Whereas the previous GML specification did not provide a means of describing visualization information, the current specification provides language constructs for the representation of default visualization styles. Given that the specification was developed with the goal of separating data and presentation, none of the data description constructs can describe visualization information. Rather, this information is given in a separate model that can be associated to a GML data set.

3.4.4 Metadata and Feature Cataloguing

The models described in the previous sections allow a user to represent and manipulate geographic information. A user can define an application schema using the conceptual model described in Section 3.4.1, implement it in a GIS application using the logical model and physical models described in Section 3.4.2, and use the language defined in Section 3.4.3 to exchange geographic information. By using these standards, GIS applications can share information easily because their data structures, operations and exchange languages are compatible.

However, in order to promote the dissemination, sharing, and use of geographic data it is indispensable that additional information is given regarding the types of features, attributes, associations and operations of a particular application schema. Following the OGC and the ISO/TC 211 standards, this additional information is given at two different levels of detail. First, a *feature catalogue* [ISO01a] documents a given application schema by describing the types of features, their operations, attributes, and relationships. And second, metadata elements [ISO03e] provide information that describes geographic information beyond cataloguing feature types.

In the following sections, we give a brief description of these two specifications.

Feature Catalogues

A feature catalogue defines the meaning of the feature types and their associated attributes, relationships and operations contained in the application schema. The ISO specification for a feature catalogue [ISO01a] defines precisely which information must be included in the feature catalogue, but does not define a representation model for the catalogue.

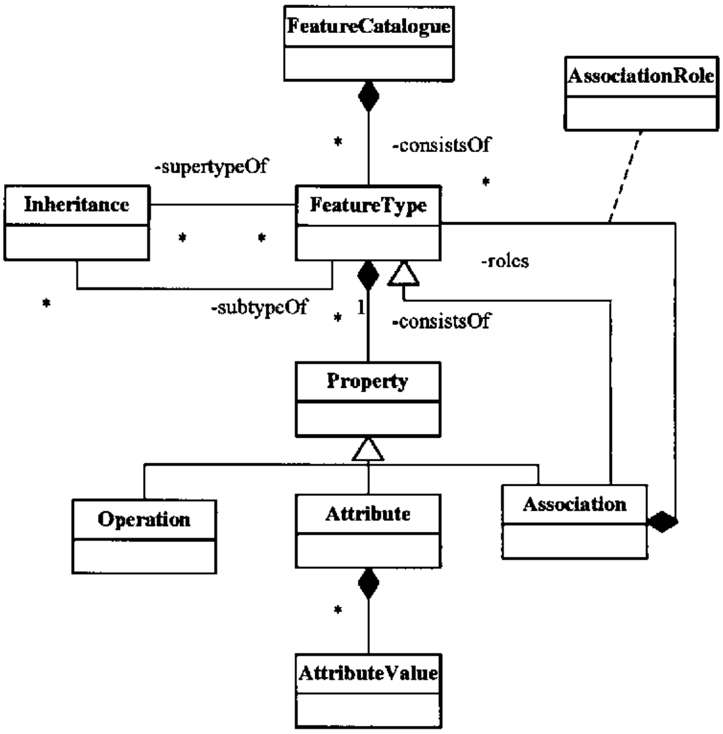


Figure 3.10: Conceptual Model For a Feature Catalogue.

Figure 3.10 shows a simplified conceptual model of the information collected in a feature catalogue. It is worth noting that the structure of the catalogue is similar to the conceptual model defined for geographic information in [ISO02e] (see Section 3.4.1). In fact, the specification details how the classes in the conceptual schema for feature catalogues implement the metaclasses defined for the conceptual model.

The utility of such a feature catalogue is two-fold. On the one hand, a user can browse the information in the feature catalogue to understand the structure of an application schema and the information contained in specific data sets that follow this schema. Similarly, a developer may use feature definitions from feature catalogues that already exist to simplify the process of developing the application schema. This will also allow to easily reuse existing data sets that conform to the feature catalogue. On the other hand, feature catalogues can be used in geographic data repositories to manage the information, or in geographic information brokers to automatically locate geographic data sets. The availability of standard feature catalogues that can be used multiple times will reduce costs of data acquisition and simplify the process of product specification for geographic data sets.

Metadata

A feature catalogue describes the structure of data by cataloguing the feature types that form an application schema. This enables a user to understand the structure of a data set that follows the feature catalogue. However, to ensure that data is not misused, the assumptions and limitations affecting the creation of data must be fully documented. The metadata elements defined in [ISO03e] enable information system analysts, program planners, developers of geographic information systems, as well as others, to describe a data set in such a detail that users can understand the assumptions and limitations, and to evaluate the applicability of the data set for their intended use.

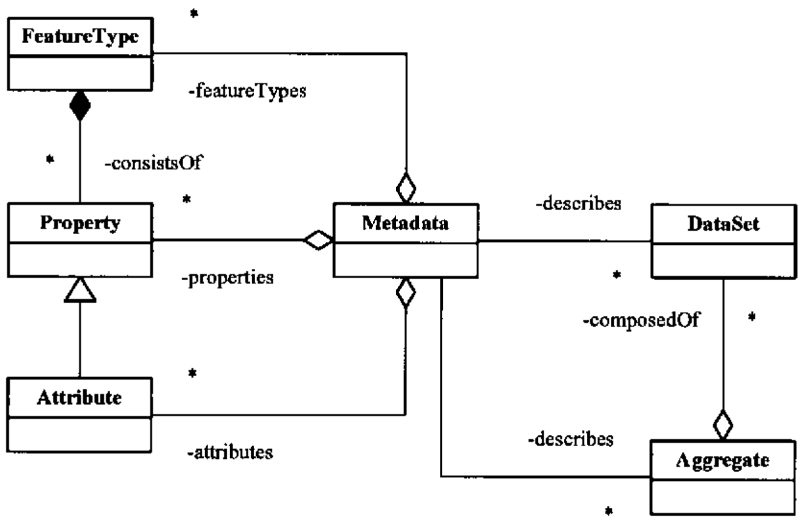


Figure 3.11: Associating Metadata to Information Elements.

The specification for metadata elements defines two kinds of conceptual entities: those that are used to represent the information elements, and those that are used to represent the metadata elements. Figure 3.11 presents a simplified conceptual model that shows how metadata elements are associated to information elements (e.g., data sets, feature types, feature properties, or feature attributes). Each of these information elements can be associated with a *metadata entity set* that defines metadata about a resource or resources and aggregates a number of other metadata elements that provide further information regarding the resources. The following categories of information elements can be associated to a collection of metadata elements:

- *A data set.*
- *An aggregation of data sets.* In this particular case , the type of aggregation can be further specified in aggregations resulting from a special initiative, from a series, of from other associations.
- *Individual geographic feature instances.*
- *Feature attributes.*
- *Feature types.*
- *Feature type properties.* That is, feature attribute types, feature associations or feature operations.

Each metadata elements represent information regarding the following categories:

- *Identification information.* It contains basic information required to uniquely identify a resource or resources, including information about the citation for the resource, an abstract, the purpose, credit, the status and points of contact. Additional metadata elements can be given to provide a description of the data format, a graphic to illustrate the data set, keywords describing the resource, constraints that apply to the resource, the frequency and the scope of resource updates, and basic information about specific applications for which the resources has been used. It is also possible to further identify the resource as being a data set or a service, and provide detailed information for each case, such as spatial representation and resolution information for data sets.
- *Constraint information.* It consists of information concerning the restrictions placed on data, including restrictions and legal prerequisites for accessing and using the resource, and handling restrictions imposed on the resource for national security or similar security concerns.
- *Data quality information.* A general assessment of the quality of the data set. It includes information about the sources and production processes used in producing a data set, and information regarding the tests applied to the data to check its quality.
- *Maintenance information.* It contains information about the scope and frequency of data updates.
- *Spatial representation information.* It consists of information concerning the mechanisms used to represent spatial information in a data set. It describes whether the representation uses a vector-based model or a tessellation-based model, and provides further information for each case. Metadata for spatial data representation are derived from [ISO03b].

- *Reference system information.* A description of the spatial and temporal reference systems used in a data set. Metadata for reference system information are derived from [ISO02d, ISO03c, ISO03d].
- *Content information.* It consists of information identifying the feature catalogue used or information describing the content of a coverage data set.
- *Portrayal catalogue information.* It consists of information identifying the portrayal catalogue used.
- *Distribution information.* It contains information about the distributor of and options for obtaining the resource. It consists of elements describing the following aspects:
  - technical means and media by which a resource is obtained from the distributor,
  - the computer language construct that specifies the representation of data objects in a record, file, message, storage device or transmission channel,
  - and common ways in which the resource may be obtained or received, and related instructions and fee information.
- *Metadata extension information.* It consists of information about user specified extensions describing the extended metadata elements.
- *Application schema information.* A description of the application schema used to build a data set.

It can be seen by the kind of information represent by the metadata conceptual model, that this information is more related to usage considerations of data sets than to formally describe the information. Nevertheless, the utility of a metadata catalogue for geographic information is similar to that of a feature catalogue. First, it can be used by a user to locate and evaluate geographic information for a particular application. And second, it can be used by computer tools to perform automatic data location, or data mining.

3.5 Data Services for Geographic Information

The specifications described in the previous sections are concerned with the definition of models and languages for the representation and manipulation of geographic information. However, none of the specifications is concerned with the definition of interaction mechanisms. A generic architecture for GIS applications cannot be built using only models for the representation of information. It is important to define software components that encapsulate parts of the functionality of different business processes. For instance, in addition to the definition of a model for the representation of geographic information, it is necessary to define a software component for data manipulation that includes operations to:

- *Create a new feature instance.*
- *Delete a feature instance.*
- *Update a feature instance.*
- *Query features based on spatial and non-spatial constraints.*

The ISO and the OGC have decided to define a highly-distributed architecture for GIS (see Section 2.6.2) by means of the definition of multiple services, each responsible for very specific functionality. Given that the evolution of distributed platforms is oriented to using HTTP as the distributed computing platform, both organizations have defined a number of a web-based services for the manipulation of geographic information. We describe in this section these services. Section 3.5.1 presents the Web Feature Service [OCG02b] that is responsible for basic data manipulation on geographic features that represent object-based geographic information. Then, Section 3.5.2 describes the Web Coverage Service [OCG03b] that provides interfaces for the retrieval of field-based geographic information.

Other specifications defined by the OGC for data manipulation are the Catalog Interface specification [OGC02a] and the Coordinate Transformation Services specification [OCG01a]. Their detailed description is outside the scope of our work. The former defines a common interface to perform discovery, browse, and query operations against distributed and potentially heterogeneous catalogues of metadata for geographic information, services and related resource information. The latter specification is related to services that transform geographic information from one coordinate reference system to another. Whereas the Catalog Interface Specification [OGC02a] is clearly part of the topmost interface of the information management tier, the Coordinate Transformation Services specification can be considered a service of the processing tier.

3.5.1 Web Feature Service

The main goal of a Web Feature Service (WFS) [OCG02b] is to offer a web-based service for querying and manipulating geographic information retrieved from any data source that implements one of the specifications for data representation and manipulation described in the previous section. According to the functionality offered by the WFS, the specification defines two different types of WFS:

- *Basic WFS.* This kind of WFS offers only read-only functionality. More precisely, it only supports data and functionality discovery, data retrieval and feature type description.
- *Transaction WFS.* In addition to all the operations supported by a basic WFS, a transaction WFS supports data manipulation operations (insert, update and delete). Furthermore, a transaction WFS may also provide serializable transactions by means supporting lock requests on the features.

The implementation specification provides a detailed description of operations, request parameters and expected responses for each of the functionality described above. Now, we give a brief description of these operations:

- *Data and functionality discovery.* Any WFS must have the ability to describe the capabilities of the service and the data that can be queried and be manipulated through the service. This is implemented by the operation *GetCapabilities*. The response to the operation includes information regarding:
  - *Service metadata.* General metadata for the service, such as a name, a title, the URL of the service, keywords and other optional information .
  - *Capability metadata.* The WFS and filter operations that are actually supported by the service, and the data formats available for encoding query results.
  - *Information metadata.* The list of feature types available in the service, and the manipulation operations (i.e., query, insert, update, delete and lock) supported for each feature types.
- *Data retrieval.* This is the most important functionality of the service, and it is implemented by the operation *GetFeature*. The response to this operation is a collection of feature instances computed according to the following arguments:
  - *A feature type name.* It indicates the name of the feature type whose instances are going to be retrieved.
  - *A list of properties to be selected from the feature type.* Only the properties enumerated by this argument are included in the response. If the list is empty, then all properties of the feature type are retrieved.
  - *A filter.* It defines constraints that must be satisfied by the feature instances to be included in the response. They are expressed using the filter language [OCG01b] described below.
  - *The output format.* It defines the format used to generate the response. All types of WFS must as least support GML [OGC02c] as output format. Other vendor specific formats may be offered (by declaring them in the response to the operation *GetCapabilities*).

- *Feature type description.* In order to enable clients to discover the structure of the feature types offered by the service, a WFS may allow clients to fetch a schema description of the feature types. This functionality is implemented by the operation *DescribeFeatureType*.

- *Data manipulation.* In addition to the previous operations, which are implemented by a Basic WFS, a Transaction WFS implements additional functionality for data transformation (i.e., create, update, and delete operations). This is implemented by the operation *Transaction*. Each invocation of this operation may include multiple transformation operations (i.e., *insert*, *update*, and *delete*), which are executed as an atomic operation by the WFS.

- *Data locking.* Given that web-based services are inherently stateless, the semantics of serializable transactions are not preserved. To enable this functionality, the specification allows clients to request a lock on a set of features for the duration of a transaction. This functionality is implemented by the operation *LockFeature* and the operation *GetFeatureWithLock*.

All arguments given to these requests are expressed using an XML application defined in the specification, and invoked using HTTP requests. Geographic features used as arguments for the operations are represented using GML. Query results are encoded at a minimum using GML, but other additional data formats are allowed for query results. The data storage for geographic features should be opaque to client applications and their only view of the data should be through the WFS interfaces.

The query constrains of a *GetFeature* operation, and the scope of the operations *Update*, *Delete* and *Lock* are represented using a filter language defined as an XML application [OCG01b], which is an extension of the OGC Common Catalog Query Language defined in [OGC02a]. This query language supports the following functionality:

- *Spatial operators.* A filter expression may include a spatial operator between a feature property and a geometry value defined using GML. The spatial operators supported are: *equals*, *disjoint*, *touches*, *within*, *overlaps*, *crosses*, *intersects*, and *contains*. The semantics of these operators is defined in [ISO00d]. Furthermore, an operator *bbox* is provided as a compact way of encoding a bounding box check. Finally, two additional operators are provided to fetch the features that are within or beyond a specified distance (i.e., *dwithin* and *beyond*).

- *Comparison operators.* This category includes the operators *isEqualTo*, *isNotEqualTo*, *isGreaterThan*, *isLessThan*, *isLessThanOrEqualTo*, *isGreaterThanOrEqualTo*, *isLike*, *isNull*, *isBetween*.

- *Logical operators.* This category comprises the operators *and*, *or*, and *not*.

- *Feature identifiers.* A single geographic feature instance can be referenced in a filter by using this operator and a unique identifier for the feature instance within the context of the service.

- *Expressions.* Expressions can be built using arithmetic operators (i.e., *add*, *sub*, *mul*, and *div*), literal values (i.e., *literal*), properties of feature instances (i.e., *propertyname*), and functions composed of a name and a collection of arguments (i.e., *function*). The operator *function* enables the developer to access all the functionality of the underlying data source, because vendor-specific function names can be used with this operator.

3.5.2 Web Coverage Service

The implementation specification for Web Feature Services (WFS) [OCG02b] is strongly oriented to query and manipulate geographic features of an object-based representation of space (see Section 2.3.2). For instance, GML [OGC02c] is the default language used to represent the data values, which does not allow to represent coverages. Moreover, the language used to specify query constrains [OCG01b] provides operators that are only applicable to an object-based view of geographic space.

In order to overcome this limitations, a Web Coverage Service (WCS) [OCG03b] allows a client to retrieve field-based geographic information using the HTTP protocol. The functionality that must be provided by a WCS comprises the following: data functionality and discovery, data retrieval, and coverage description. This functionality is implemented by a set of operations defined in the specification, which also provides a detailed description of the arguments that are passed to the operations, and the response expected from each operation. Now, we describe briefly each of the operations defined in the specification:

- *Data and functionality discovery.* Each WCS must describe its capabilities. This information is provided by the operation *GetCapabilities*, which return a response including the following:

  - *Service metadata.* General metadata for the service, such as a name, a title, the URL of the service, access fees or keywords.

  - *Capability metadata.* It describes the requests that the service supports, the formats in which exceptions are returned, and any other vendor-specific service capabilities.

  - *Content metadata.* In addition to giving (optional) descriptions of the coverages offered by the service, the response may point to another source of metadata for the content of the service, such as an image catalogue service. This is intended to be used by services that offer a large number of coverage. In these services, if each coverage were described by metadata elements, then the size of the response to this operation would be too large to be fetched over the Internet. Instead, this type of services must point to a catalogue describing the service contents where searches for specific coverages can be performed efficiently.

- *Coverage description.* A client must be able to request and obtain a full description of any of the coverages offered by the WCS. This functionality is implemented by the operation *DescribeCoverage.* The response to this request includes the following information for the coverages:
  - *Name, title and description.* Textual labels and descriptions for the coverage.
  - *Bounding box.* Envelope of the geographic space covered by the coverage.
  - *Domain and range set description.* Detailed description of the domain (space covered by the coverage) and the range (attribute values of the coverage).
  - *Supported coordinate reference systems.* Coordinate reference systems in which the service understand data retrieval requests for the coverage.
  - *Supported output formats.* Advertises the formats in which coverages may be requested from the service.
- *Data retrieval.* A WCS must be able to service a coverage request issued by a client. This functionality is implemented by the operation *GetFeature.* In addition to the name of the coverage, other parameters can be used to specify the bounding box of the requested coverage, the desired output format and output coordinate reference system, and additional parameters to further specify the coverage requested.

3.6 Portrayal of Geographic Information

The models, services, and languages defined in the previous sections are in charge of the representation and the manipulation of geographic information at various levels of abstraction. For instance, the simple feature geometry model (see Section 3.4.2) deals with the representation of geometry values in a computer system at the level of the logical model, and the Web Feature Service implementation specification (see Section 3.5.1) defines a service for manipulating geographic information at the level of the information management tier. Other services (not yet defined by the OGC nor the ISO/TC 211), might provide for manipulation of geographic information at the level of the processing tier (e.g., a route-finding service, or a remote detection service in satellite images).

On top of these services (i.e., in the human interaction tier), there must be services that convert the geographic information into display elements that can be visualized in a display device and manipulated by the end-user of the GIS application. For this task, the OGC currently defines a Web Map Service (WMS) [OCG02c] that produces a visual representation of geographic information. The specification is not concerned with the retrieval of actual feature data or coverage data values, only with its rendering in a graphic format such as PNG or SVG.

When this service is used to request a map, a client may specify the information to be shown on the map (i.e., the *layers*), and the visual representation of each feature type in the map (i.e., the *styles*). According to the way in which layers and styles are defined and specified in the request, the specification defines two different kinds of services: a *basic WMS* and a *Styled-Layer Descriptor WMS* (SLD-enabled WMS). In a basic WMS, a developer defines the set layers and the set of named styles for these layers that are available in the service. The client of the WMS is limited to selecting a subset from these layers and applying one of the named styles to each layer. There is no possibility for the client to define new layers or styles, or to check the style definitions. On the other hand, in an SLD-enabled WMS, the layers and theirs styles available in the WMS are defined using the SLD language [OCG02a]. This kind of WMS allows a client to define new layers and styles in addition to those predefined in the WMS.

In the following sections, we briefly describe each kind of WMS. First, we show the functionality of a basic WMS in section 3.6.1. Then, we present in Section 3.6.2 the extensions to this kind of WMS provided by an SLD-enabled WMS.

3.6.1 Basic Web Map Service

The functionality that must be provided by a basic WMS comprises the following: data functionality and discovery, map production, and feature information retrieval. Geographic information in a WMS is structured as a set of *layers*. Each layer may contain information from different data sources and data types, and may have different graphical styles applied to the geographic features. However, a layer is the basic query unit of a WMS and must be retrieved as such.

Given that the service is intended to operate in a web-based environment, the specification defines a set of operations that implement the aforementioned functionality, and defines the parameters and responses expected from each operation. Now, we briefly describe each of the operations defined in the specification:

- *Data and functionality discovery.* In order to enable clients to operate and query the WMS, it is necessary that the service provides general information about the service itself and specific information about the available geographic information. This is implemented by the operation *GetCapabilities*. No specific parameters are required by calls to this operations. The response to the operation includes:

  - *Service metadata.* This information includes general metadata for the service, such as a name, a title, the URL of the service, and other optional information (e.g., contact information, or access fees).
  - *Capability metadata.* The operations and the output formats that are actually supported by the service are listed in this information.
  - *Information metadata.* Finally, the response to the operation includes the list of available layers. For each layer not only its name is given, but also its geographic extent, spatial reference system, and other information. More importantly, each layer is associated to a collection of display styles that are applicable to the layer.

- *Geographic information portrayal.* The most important functionality that must be provided by the service is the production of a map from a set of geographic features. This is implemented by the operation *GetMap* by specifying the following parameters:

  - The information to be presented as a set of layers.
  - The visual representation for the information as a set of styles.
  - The coordinate reference system used for the projection.
  - The geographic extent of the map as a bounding box. Furthermore, the specification defines optional parameters that can be used to indicate additional dimensions (e.g., time or elevation).
  - The size, storage format and background color of the output map.

  The response of this operation is a graphical representation of the requested geographic information in the format specified in the parameters. When two maps are requested with the same geographic extent, coordinate reference system, output size, and transparent background, they can be accurately overlaid to produce a composite map. This enables clients and end-users to compose maps with geographic information fetched from different, independent sources.

- *Retrieve information of the features in the map.* A client can request further information from a feature that has been displayed in a map by specifying a point in a map (i.e., the WMS solves *point queries*). This functionality is implemented by the operation *GetFeatureInfo*, which receives the following information as parameters:

  - The arguments that were used to compose the map that is being queried.
  - The maps layers for which additional information is being requested.

– A point in the map.

The standard does not specify the nature of the response, that is, the format and the content of the response is particular to each WMS. The specification only requires that the response is related to the features nearest to the queried point.

3.6.2 Styled-Layer Descriptor Web Map Service

Whereas a Basic WMS is an important step towards a portrayal service for geographic information, its functionality is not sufficient to fulfill the client requirements. While a Basic WMS provides the client with a choice of styles for each layer, the client only knows the name of the style, but cannot preview what the style will look like on the map. Additionally, in a Basic WMS there is no standard way of defining styles, and furthermore, clients cannot provide their own styling rules for the information.

To overcome these limitations, the OGC has defined in [OCG02a] an implementation specification for a language and a set of extensions to the Basic WMS. The language is called the *Styled Layer Descriptor language*, and a WMS supporting this language and the extensions defined in the specifications is called an *SLD-enabled WMS*.

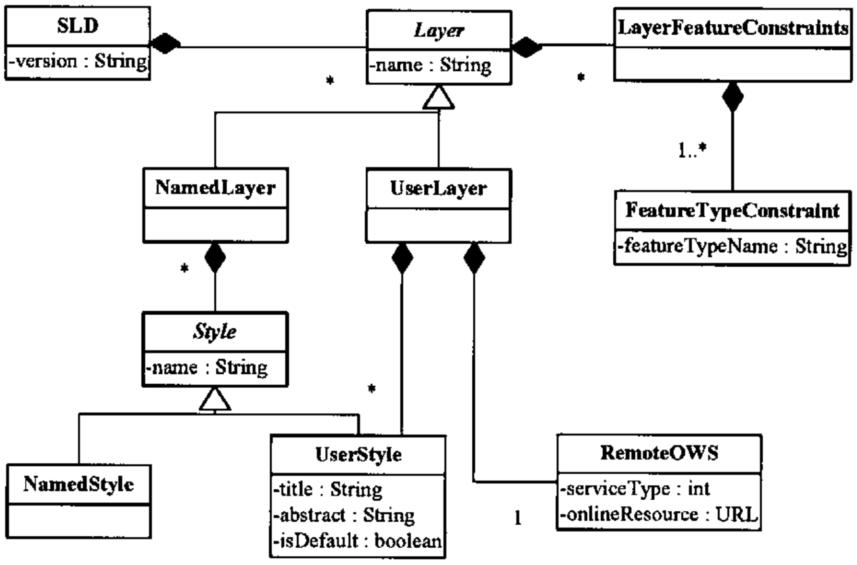


Figure 3.12: Conceptual Model for SLD.

Figure 3.12 shows a simplified conceptual model of the SLD language. An SLD file consists of a collection of layers represented by the abstract class *Layer*. Two types of layers can be defined using SLD: *named layers* and *user layers*. A named layer references a layer defined internally by the WMS, and a user layer consists of a layer defined by the user. For both types of layers, a collection of layer feature constraints can be given to further specify the geographic features to be displayed. For the named layer, it is used as a filter on the layer defined by the WMS. On the other hand, for a user layer it is used as the definition of the data that has to be fetched from the remote web service specified in the *RemoteOWS* object associated to the *UserStyle* object.

Similarly to layers, there are two types of styles (represented as subtypes of the abstract class *Style*): *named styles* and *user-defined styles*. Named styles are defined internally by the WMS, and can be applied only to named layers. On the other hand, user-defined styles are completely defined by the user using the SLD language and can be applied to both types of layers.

Figure 3.13 shows a simplified conceptual model of the style-definition section of the SLD language. First, a style consists of a collection of style-definitions for each feature type that contributes geographic features to the layer. Then, for each feature type, a collection of *painting rules* are given. Each rule is assigned a *filter* that selects the features that must be painted by the rule, and a *symbolizer* that defines the graphical characteristics of the geographic features.

An SLD-enabled WMS can be used in many different ways, which are a mixture of the following operation modes:

- *Using the SLD language as a style library.* An SLD-enabled WMS defines the styles for the layers available in the service using the SLD language. Clients use this WMS just like a Basic WMS. However, clients can also query the definition of the styles, preview the portrayal result, and developers can use the SLD language to define the styles.
- *Using the SLD language to specify user-defined styles.* In addition to using named styles defined by the WMS, a client can also provide user-defined styles in the portrayal request This gives the client more freedom for the portrayal process. Furthermore, an SLD-enabled WMS may allow clients to add newly-defined styles to the style library, thus working as a shared style repository.
- *Using the SLD-enabled WMS as a component portrayal service.* The SLD language allows not only to define new styles, but also to define new layers for the WMS. This enables clients to use the SLD-enabled WMS as a painting machine for geographic data that may come from different sources.

An SLD-enabled WMS implements the following functionality:

- *Data and functionality discovery.* In addition to the service metadata provided by a Basic WMS, an SLD-enabled WMS must provide clients with the following information:

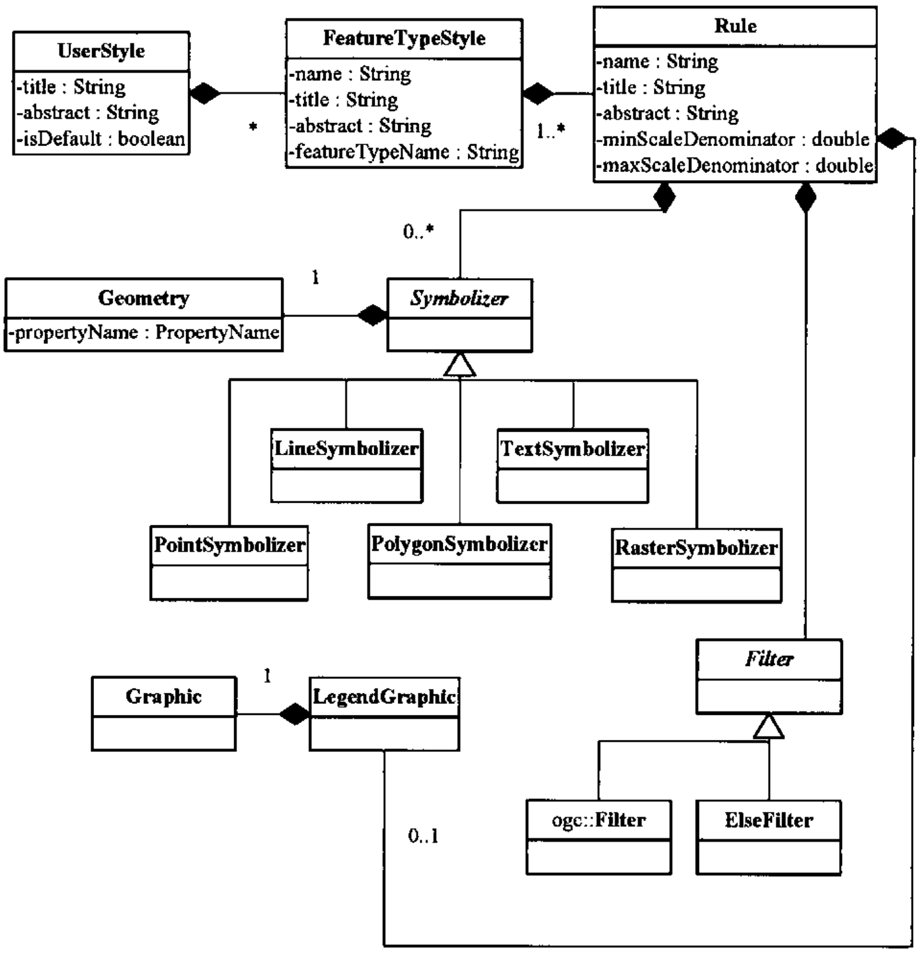


Figure 3.13: Definition of a User Style in SLD.

  - *SLD support.* The WMS must indicate whether it is an SLD-enabled WMS.
  - *Support for user-defined layers.* Indicates whether users can define their own layers in addition to those offered by the WMS.
  - *Support for remote data services.* Indicates whether a remote WFS or WCS can be used as the data source for user-defined layers.
  - *Support for user-defined styles.* Indicates whether users can define their own styles in addition to those offered by the WMS.

- *Geographic information portrayal.* In addition to the parameters defined for a Basic WMS, an SLD-enabled WMS must provide parameters to indicate the SLD file used for the portrayal.

- *Layer description.* In order to enable a client to define user-defined styles, the WMS must provide a means for inspecting the definition of the layers defined by the WMS. This is implemented by the operation *DescribeLayer.*

- *Graphical legend generation.* Given that in a SLD-enabled WMS the style definitions are not fixed, the graphical legend for a map can no longer be predefined. Instead of delegating the responsibility of generating the graphical legend on the client, the specification provides an operation to generate icons for the map styles. This operation is called *GetLegendGraphic.*

- *Style-library management.* An SLD-enabled WMS stores the styles in a style-library that can be (optionally) queried and extended by clients. This functionality is implemented by the operations *GetStyles* and *PutStyles.*

3.7 An Analysis of the OGC and ISO Proposals

After giving a description of the OGC and the ISO/TC 211 proposals, we are going to analyze them in order to point out their strengths and their drawbacks, which may motivate a new or complementary proposal.

The OpenGIS consortium and the ISO/TC 211 are carrying out a very important task in the field of geographic information systems. They are laying the foundation for a new generation of GIS applications and development tools that will be able to cooperate to greater extent at many different levels.

Furthermore, instead of proposing a monolithic software layer implementing all the functionality of GIS applications, these organizations propose to break down the functionality in a vast collection of services with very specific functionality that interact only at the interfaces. This will allow a very flexible GIS application architecture because the services implementing the functionality may be running in a single computer, or distributed along a wide-area network, in a totally transparent way.

However, the main drawback of the architecture proposed by these organizations is that there are many pieces still missing. Even though the information management tier is almost completely defined, no specifications have been defined for the processing tier, and very little work has been produced for the human interface tier.

We have depicted in Figure 3.14 the architecture of a GIS applications with two different user interfaces, which is built over specifications already published. The first user interface is a desktop-based GIS for geographic data analysis, whereas the second one is a web-based GIS for geographic information portrayal. The figure shows the specifications that are already adopted in white boxes, and the missing pieces of the architecture in gray boxes. It can be seen that the specifications defined by the OGC and the ISO/TC 211 provide a complete information management tier, but much of the processing algorithms and information portrayal techniques must be implemented by a developer.

Another drawback is that the intrinsic nature of any standards organization causes the process of developing a specification to be rather slow. Before a specification is adopted, it must be proposed, written in a draft state, discussed and voted by the membership. This is a lengthy process that cannot be assumed by software companies that must produce novel products at a much faster pace.

Moreover, the definition of specifications by multiple groups working independently may cause that the specifications do not match perfectly due to coordination problems. Two concrete examples of this problem are shown in the following list:

- *Duplicate definitions.* Both the specification for GML [OCG03a] and the SLD language [OCG02a] provide a definition for constructs to specify the visual style of a cartographic object generated from a geographic feature. This is a problem because it duplicates the definition of constructs for the same purpose in two different specifications. Moreover, the definition of these constructs should not be part of the GML, because it is a language for the representation of geographic features, not for the portrayal.

- *Incomplete definitions.* When a service is defined, not all the functionality of an underlying service or specification is offered. For instance, the Web Feature Service [OCG02b] is defined as a service for querying and manipulating geographic features. However, the specification only supports the object-based view of geographic space (i.e., geographic features with attributes of a spatial data type). Information represented using the field-based view of geographic space (i.e., coverages, which are a special type of geographic feature) must be queried using a completely different service with a completely different interface. Even though the conceptual model defined for the information management tier (see Section 3.4.1) offers a single abstraction for geographic real-world phenomena (i.e., a feature), object-based and field-based

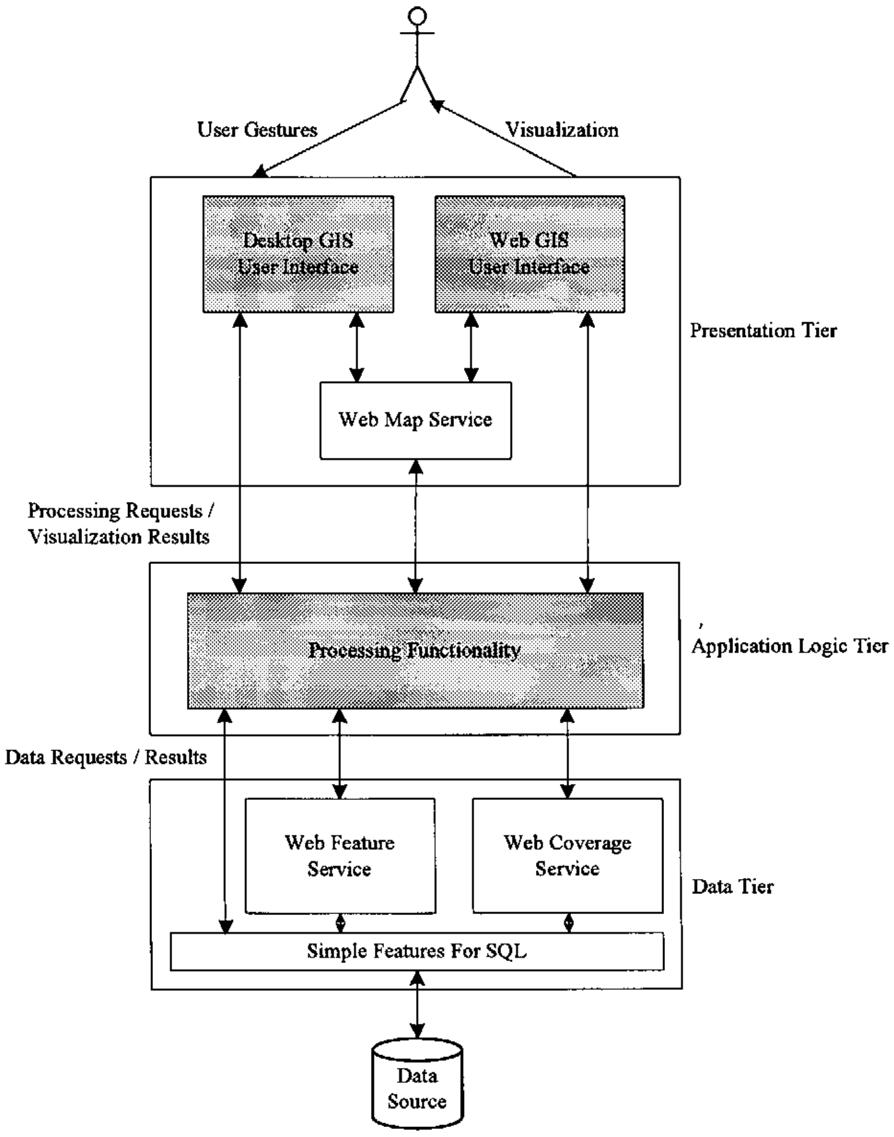


Figure 3.14: OGC Services in the Architecture of a GIS.

representations of space must be queried and be manipulated using different interfaces (i.e., the Web Feature Service and the Web Coverage Service), and later mixed at the human interface tier (by means of a Web Map Service).

Another example is the limitation that the WFS imposes on the types of queries that can be posed over the data source. Instead of providing a complete query language like the one defined for the simple feature geometry model [ISO00d], a very limited query language is offered by the specification. More precisely, the language does not support relational joins, function applications to compute new values in the query result, or the complete set of spatial operators.

A final problem of the standards defined by these organizations is caused by the extensive use of XML-based technologies for the definition of exchange languages. Obviously, the use of XML applications is motivated by their advantages:

- XML provides an open, vendor-neutral framework for the representation of data.
- XML Schema allows developers to define application schemas for a particular application domain using and extending the definition provided in the specification.

However, these languages have also some important drawbacks. The major one is that the representation of a feature is very verbose and the ratio between information content and file size is very low. Moreover, given that XML is a markup language, the data structure is mixed with the data contents.

3.8 Summary

In this chapter, we have presented the main standard specifications proposed by the OpenGIS Consortium and the ISO Technical Committee 211 for the representation, manipulation, and visualization of geographic information. The work that is being developed by these organization is of great value, and it must be considered as a starting point for the development of any GIS application.

We have also analyzed the drawbacks of these specifications. The main problem of is that there are still many missing pieces in the architecture because the proposal is far from being complete. Therefore, it is necessary to design many more software components to have a complete view of a generic architecture for GIS. Of course, the new proposal must conform to the OGC and the ISO/TC 211 standard specifications where possible. The following chapter presents a proposal for a generic architecture for GIS applications that takes into account the special characteristics of geographic information presented in Chapter 2 and conforms with the specifications presented in this chapter.

Chapter 4

A Generic Architecture for Geographic Information Systems

4.1 Introduction

We have shown in Chapter 2 that the special nature of geographic information imposes special requirements on the architecture of GIS applications with respect to the architecture of general-purpose information systems. The OpenGIS Consortium and the ISO/TC 211 make much effort to define a generic architecture for GIS applications that meets these requirements, as was described in Chapter 3. Even though the value of the specifications produced by these organizations cannot be denied, the development of a real-world GIS application requires the definition of a complete architecture that has not yet been finished by those organizations, as we have already pointed out.

In this chapter, we present a proposal for a generic architecture for GIS that defines a complete framework for the development of GIS applications. We describe a collection of generic components and their interaction mechanisms that allow a developer to build a GIS application by selecting, connecting, and customizing a subset of these components. The architecture design is based on the analysis of the requirements imposed by the special characteristics of geographic information of Chapter 2. Moreover, our proposal conforms with the OGC and the ISO/TC 211 specifications wherever standards have been published, and proposes new components wherever no standardization work has yet been accomplished.

The rest of this chapter is structured as follows. In Section 4.2, we enumerate and describe briefly the requirements for the architecture of a GIS application, which are derived from the special characteristics of geographic information and the requirements for general-purpose information systems. Then, we propose in Section 4.3 a generic architecture for GIS applications that meets these requirements.

In the sections that follow, we describe the components of this architecture in more detail, and we illustrate its functionality through existing standards, commercial applications, or research results. Section 4.4 describes the components related to information management, Section 4.5 presents the components related to application-specific functionality, and Section 4.6 details the components that deal with the user interface of the system. In addition to describing the architecture in detail, it is necessary to show how specific applications can be built using this architecture. This is done in Section 4.7. Moreover, we also present in this section a framework to generate GIS applications in a semi-automatic way.

Finally, in Section 4.8, we summarize the chapter and we compare and analyze our proposals with respect to the standards proposed by the OGC and the ISO/TC 211.

4.2 Requirements for the Architecture of a Geographic Information System

Many years of research in the field of general-purpose information systems have produced a set of requirements that must be met by any architecture proposal. On the other hand, the special nature of geographic information results in a set of characteristics of geographic information that must be taken into account in the development of a GIS application. We have analyzed these characteristics in detail in Chapter 2, and we have found out a collection of requirements for GIS development tools that we summarize in this section.

- *Flexibility.* The functional requirements and the runtime environment of an information system often change during its lifetime. Therefore, it is important that the system can adapt easily to these changes. Furthermore, it is desirable that the information system can be used in different technological platforms with different functionality (e.g., personal computers, mobile devices). Hence, the architecture and the functionality of the information system must be easily adaptable to these platforms (see Section 2.6.2).

- *Extensibility.* In addition to changes to the functionality or runtime environment, during the lifetime of an information system it is often necessary to support and incorporate new requirements and technological advances. The information system must provide a highly customizable framework that can be extended with new features by mean of a programming language or additional software components (see Section 2.8).

- *Reusability.* The development process of an information system is costly. Therefore, it is important that the components of the information system can be used again without significant modification as a building block in a different information system from the one that it was originally designed for. This requires the development of generic modules that can be configured for specific tasks by means of high-level languages (see Section 2.6.2).

- *Scalability.* Even though it is possible to estimate the number of users that will use the information system or the computing power needed by the functionality in the information system, these estimations are likely to change during the lifetime of the system. Hence, the system must be designed in a way that it is possible to increase the number of users or expand the capabilities of a computing solution without making major changes to the system (see Section 2.6.2).

- *Reliability and security.* Information systems need to be fault-tolerant and highly-available. Moreover, it has been repeatedly proven that security is a very important requirement for computer-based information systems.

- *A conceptual model for geographic information.* It is necessary to define a conceptual model that supports the description of geographic information. The model must include abstractions to model real-world geographic phenomena, using both the object-based and the field-based views of space (see Section 2.3.2).

- *Primitive operations on the abstractions of the conceptual model.* An exhaustive set of primitive operations on the data abstractions of the conceptual model must be defined in order to support data querying (see Section 2.4.1).

- *A query language for the conceptual model.* This language is used to retrieve and manipulate the data abstractions of an application schema defined using the conceptual model. The language must be composed by the operations defined previously. The results of queries must be represented by an appropriate language that captures all the abstractions of the conceptual model.

- *Problem-solving techniques.* The abstractions in the conceptual model and the query language must be used to provide generic solutions for common geographic problems. There must be a precise definition of the problems to be solved, the information needed, and the techniques used to solve them (Section 2.4.2).

- *Metadata and data cataloguing.* In order to define generic methods to find out and use the information contained in any given application schema, it is necessary to provide models for metadata and data cataloguing (Section 2.7).

- *Logical models for specific technological platforms.* Given that each technological platform has particular limitations for the representation of information, it is necessary to define a logical model for the platform that takes into account these limitations. This model defines data structures for the data abstractions and algorithms for the operations that can be implemented in the technological platform (see Section 2.3.3).

- *Efficient physical models.* Even though the logical model takes into account the limitations of computer systems with respect to the representation of information, it is necessary to define new storage techniques and access methods in order to achieve computational efficiency. (see Section 2.4.3).

- *Visualization and interaction metaphors.* There is a need for appropriate metaphors to manipulate geographic information at the user interface of the system. These metaphors must be based on the well-known map metaphor, and must incorporate dynamic operations such as *zoom*, *pan* and the addition and removal of information grouped in cartographic layers (Section 2.5.1).

- *Abstractions for information visualization.* The data abstractions used for the storage and processing of geographic information are not suitable for its presentation to the user. Therefore, new data abstractions are needed that support displaying different views of a single geographic object at different resolutions or with different visual styles according to display parameters such as scale (Section 2.5).

4.3 Architecture Overview

4.3.1 Description of the Architecture

Our proposal for a generic architecture for geographic information systems, based on the proposal of the ISO and the OGC (Chapter 3), is shown in Figure 4.1. The specifications defined by these organizations are used where possible. However, we provide more detail in the definition of the components in the architecture by extending the existing specifications and proposing new ones where no specification by the OGC and the ISO/TC 211 exists.

We propose an architecture that separates the functionality of the system in three independent tiers, namely the *Data Tier*, the *Application Logic Tier* and the *Presentation Tier*. In order to enable reusability and flexibility of the system architecture, the functionality of these tiers must be implemented independently of any particular application. The strategy followed finds and isolates the characteristics and the functionality that are independent from the application schema from the dependent ones. Then, the independent functionality and characteristics are implemented using

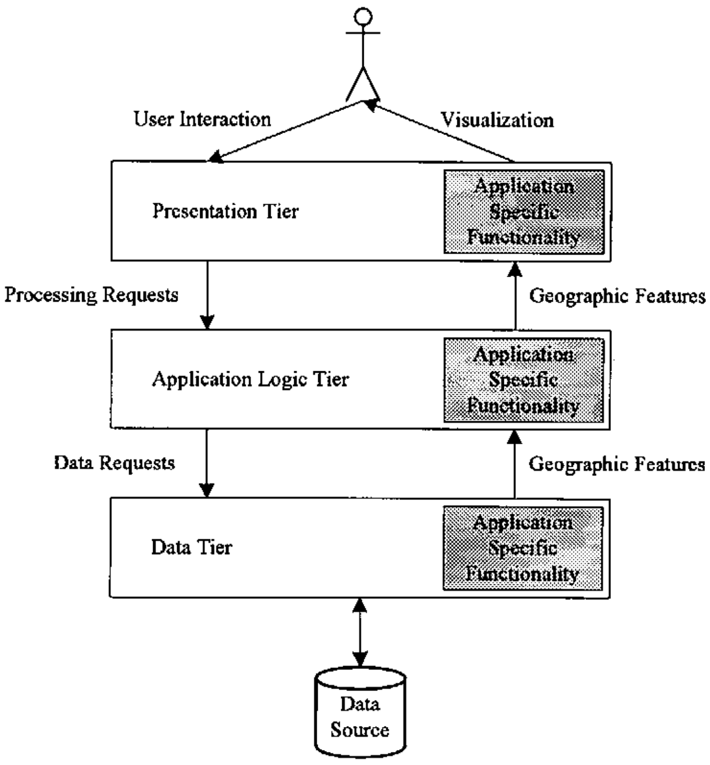


Figure 4.1: A Generic System Architecture for GIS.

generic algorithms and a developer fills in the application-specific details. This is represented in Figure 4.1 by the grayed modules in each tier, which comprise the functionality specific to a particular application. Now, we describe briefly the responsibility of each of these tiers:

- *Data tier.* It provides data management functionality independently from the software technology. Information retrieval and manipulation requests for the data tier are expressed using a query language. Queries are evaluated within the data tier and the result is a set of geographic features that are represented using an information exchange language that is suitable for the conceptual model.
- *Application logic tier.* It implements the problem-solving and the application-specific functionality of the system. The top-most interface of this tier consists of a collection of operations for data processing tasks. These operations represent high-level abstractions of problem-solving techniques (e.g., find a route between two nodes in a network) instead of the primitive operations from the data tier query language (e.g., find whether there is an direct edge between those two nodes). A processing request to this tier is expressed as an invocation to one of these operations, which is then executed by the application logic tier either by using its internal information or by building and issuing the appropriate queries to the data tier and manipulating the returned data.
- *Presentation tier.* It implements the user interface of the system, which enables data visualization, data manipulation and data entry. The presentation tier receives the user interaction in the form of mouse gestures, keyboard inputs or inputs from other devices. These inputs are evaluated and the appropriate operations in the application logic tier are invoked. When the results are returned, they are displayed to the user using the appropriate user interface controls and visualization metaphors.

4.3.2 Internal Architecture of the Tiers

Each of these three tiers is not a monolithic software module. Instead, each tier is composed by a set of software layers that divide the functionality of the tier into independent components. Figure 4.2 shows the internal layers of the tiers in the architecture. Now, we describe briefly the internal components of each of the tiers:

- *Data tier.* We propose that this tier must be internally divided into three independent layers: the *data services* layer, the *mediator* layer, and the *data sources* layer. Considering that there may be many different types of data sources, the internal architecture of the tier must be organized in a mediator-wrapper pattern. The mediator layer must offer a single conceptual model for

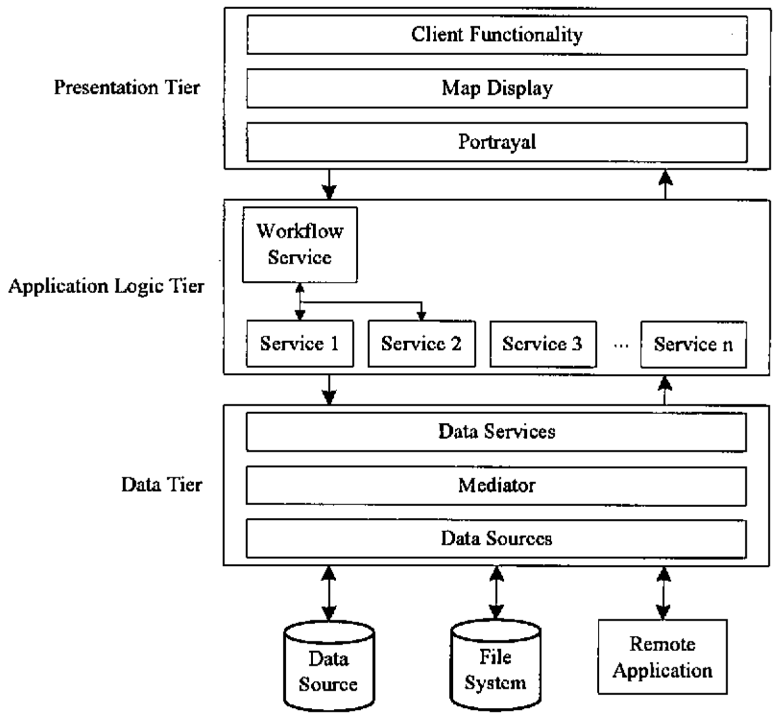


Figure 4.2: Software Layers in the Architecture.

geographic information including data types, a query language, metadata and catalogue information. Then, there is a wrapper in the *data sources* layer for each data source type that deals with the peculiarities of the data source. Finally, the data services layer provides a collection of profiles of the conceptual model for specific applications that enable to hide the complexity of the underlying conceptual model (e.g., by providing a simplified query language or a specific exchange language).

- *Application logic tier.* This tier is not divided into layers like the data tier. Instead, we propose that the functionality of the tier must be implemented by a set of multiple independent *services*. Each service performs a well-defined and simple task, and it is defined by giving its interface as a set of operations and a description of the results. The services existing in the architecture cannot be predefined because the functionality necessary for specific GIS applications cannot be known in advance. We can foresee the need for some services such as a route-finding service in networks, a geographic value simplification service, a coordinate transformation service, or an application schema transformation service. In order to achieve more complex tasks, multiple services must be chained by means of a *workflow service*, which allows developers and end-users to build a new service by connecting a collection of simple services.

- Presentation tier. Finally, we propose that this tier must be divided into three layers: the *client functionality* layer, the *map display* layer and the *portrayal* layer. We describe each layer separately:

  - *Portrayal layer.* This layer is in charge of converting a collection of geographic features into a collection of cartographic objects that can be rendered on a display device. The portrayal process is controlled by a set of styles definitions, which must define precisely the way in which each geographic feature must be rendered.
  - *Map display layer.* It is responsible for rendering the cartographic objects on the display device and enabling the end-user to manipulate and interact with the cartographic objects to perform geographic operations and to request processing operations from the application logic tier. The definition of the actions associated to each of the user-interface events is represented as a collection of activity rules.
  - *Client functionality layer.* Finally, this layer allows for the implementation of the user interface functionality. Additionally, some basic geographic functionality can be implemented in this layer using the cartographic objects for the computations (e.g., client-side map zooms, or measurements). This avoids long processing times involving server queries for simple operations.

4.3.3 Characteristics of the Architecture

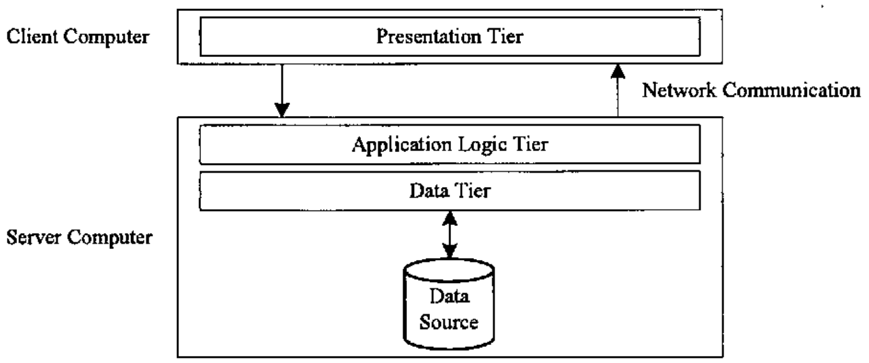
The architecture that we propose in this Chapter is *conceptual* in the sense that it can be put into practice using many different implementations. As an example, Figure 4.3 shows two different implementations of this architecture. Figure 4.3(a) shows a traditional client/server architecture. The client contains the presentation tier whereas the server contains both the application logic tier and the data tier. On the other hand, Figure 4.3(b) shows an example of a highly-distributed architecture. The presentation tier is implemented by a client computer, and the data tier is implemented by a server computer. However, the services of the application logic tier are distributed among multiple computers. Many other variations are possible.

Our architecture proposal meets the requirements that we have enumerated in Section 4.2. The architecture is *flexible* to changes in the functional requirements because of the separation of data management functionality from the application logic, and the application logic functionality from the presentation logic. Moreover, using independent services oriented to simple and specific tasks in the application logic tier causes that changes in functional requirements do not require complex software changes. Furthermore, the architecture can be adapted easily to multiple runtime environments because it is highly-modular, as it was shown in Figure 4.3.

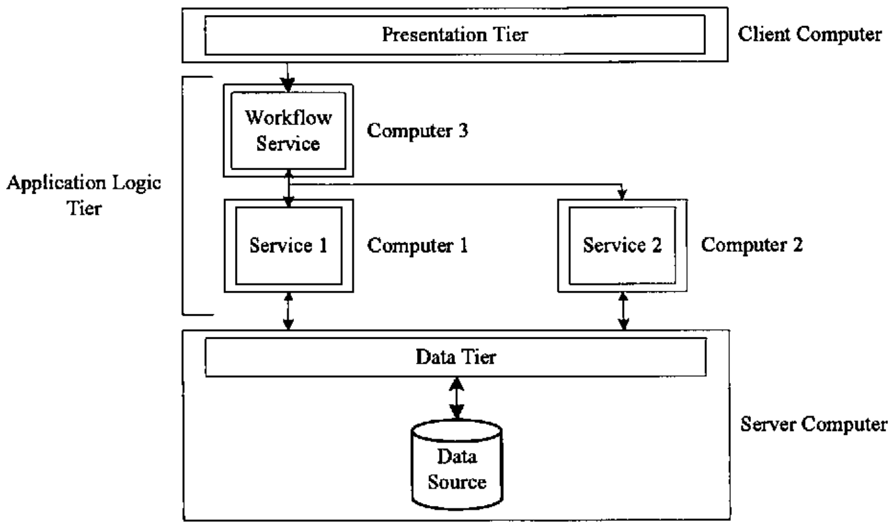
The separation of the application functionality into multiple tiers and modules within each tier makes it easier to implement *reusable* components. Moreover, new functionality can be easily added to the system using new modules for all the tiers, and thus, the architecture provides for high *extensibility*). Regarding *scalability*, the high-modularity of the three-tier architecture allows distribution of application components across multiple servers thus making the system much more scalable. Finally, this architecture makes it easier to increase *reliability* by implementing redundancy at each tier.

It is very difficult to decide which functionality belongs to the application logic tier and which belongs to the data tier. DBMS vendors include more functionality in their products with each release, and tasks that were performed by services in the application logic tier are now carried out by the data tier. However, in order to provide a clear separation of functionality in the architecture, we have determined that the functionality in the data tier must consist of primitive operations oriented to solve multiple problems, whereas the functionality in the application logic tier must be oriented to solve complex and specific problems. For instance, operations to perform basic point-set operations on geographic values, or predicates to check topological relationships between geographic values belong to the data tier. On the other hand, a service that determines the optimal route given a network and two points in the network, or a service that solves allocation problems belong to the application logic tier.

The separation of the functionality of the application logic tier in multiple independent services may affect the overall efficiency of the system. A monolithic



(a) Client/Server Architecture.



(b) Distributed Architecture.

Figure 4.3: Different Implementations of the Generic Architecture.

system can be optimized to perform some tasks efficiently, but the resulting system is barely flexible. On the other hand, a service-based architecture is highly flexible and performs efficiently each individual task, but the overall performance of a complex chaining of services is very difficult to optimize because of the inherent independence of the services. However, we believe that the benefits of service-based architecture outweigh its drawbacks, and therefore, we have chosen this type of architecture.

4.4 Components of the Data Tier

As we have already said, the goal of the data tier is to provide information management services independently from the storage technology. In order to achieve this goal, we propose that the functionality of the tier must be divided into three independent software layers: the *data services* layer, the *mediator* layer, and the *data sources* layer. Figure 4.4 shows with more detail the architecture of the data tier. The core of the tier is the *mediator* layer, which defines a single conceptual model for geographic information including geographic data types, a query language, metadata and catalogue information. Then, the *data sources* layer is used to implement a mediator-wrapper architecture for data access. For each different data source type, a component must be implemented in the data source layer that adapts the peculiarities of the data source to the conceptual model defined in the mediator layer. Finally, on top of the mediator layer, the *data services* layer allows to define specific profiles of the conceptual model defined in the *mediator* layer for specific applications. We describe these software layers with more detail in the following sections.

4.4.1 The Mediator Layer

The core of the data tier is the mediator layer. The goal of the mediator layer is offering the tools for representing geographic information, including data types, a query language, metadata and catalogue information. This layer must provide the following components:

- *A conceptual model for geographic information.*
- *A query language.*
- *A result exchange language.*
- *Metadata and data cataloguing facilities.*

In the following sections, we describe each of these components independently.

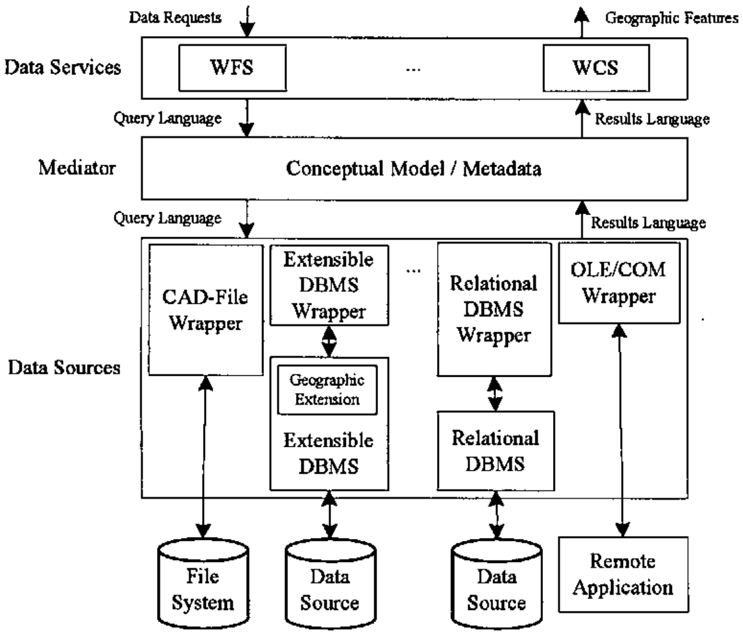


Figure 4.4: Architecture of the Data Tier.

Conceptual Model

The conceptual model of the architecture enables the end-user to describe the schema of the specific application for which the data tier is being used. The application schema provides a formal description of the content and structure of the information in the application, and a formal specification of the operations for manipulating and processing this information.

There are many proposals of conceptual models for geographic information in the literature. It is outside the scope of our work to select a particular one for our architecture proposal. Instead, we have described extensively the requirements of a conceptual model for geographic information in Section 2.3.2. Nevertheless, a strong candidate is the General Feature Model proposed by the ISO/TC 211 in [ISO02c], which will be probably adopted by the OGC as well. This conceptual model delegates on [ISO03b] the description of a conceptual model for geographic information in an object-based view of geographic space.

With respect to our proposal, it must be important to understand that the capabilities of the conceptual model chosen for the mediator layer determines the functionality available at all levels of the architecture. For instance, if there is a poor support for geographic operations in the conceptual model chosen, then any additional operations must be implemented in components of the data services layer, the application logic tier or the presentation tier, thus resulting in reduced efficiency.

Query Language

Requests to the mediator layer must be expressed using a query language. This language must be defined after the conceptual model, because the query language must support all data abstractions and manipulation operations defined in the conceptual model. Particularly, it must support:

- *Data Querying.* Retrieving data that fulfills some conditions and computation of new values using the data abstractions and operations defined in the conceptual model.
- *Data Manipulation.* Creating, deleting and modifying values of the data abstractions.
- *Data Definition.* Creating, deleting and modifying elements of the application schema.
- *Security Management.* The language must allow to define permissions for data access and manipulation.

Two different kinds of query language may be defined:

- *A high-level, text-based query language.* Queries are expressed using a declarative text-based language (e.g., any of the multiple SQL extensions for spatial data querying). Examples of this type of language are the SQL extension defined by the OGC in [OGC99b, ISO00d] or the XML-based language defined by the OGC to retrieve data from a Web Feature Service [OCG02b, OCG01b].

- *A low level, API-based query language.* Queries are expressed using objects and operations of a programming interface. Two examples of this type are the OLE/COM approach defined by the OGC in [OGC99a] and the CORBA approach defined by the OGC in [OGC98].

The advantages and drawbacks of each kind of language are complementary. Text-based languages are user-friendly whereas API-based languages need expertise in programming languages. On the other hand, text-based languages need to be wrapped by an API to be used from a programming language, which is obviously not necessary for API-based languages. Nevertheless, in order to define a flexible architecture, both types of languages must be available.

Furthermore, each of the text-based languages proposed has its own advantages and drawbacks:

- SQL extensions [OGC99b, ISO00d] must be backward compatible with previous versions of SQL, which means that the relational background of SQL is still present in the language. This implies that querying an object-oriented conceptual model such as the general feature model [ISO02e] results in awkward queries that are difficult to formulate for complex retrieval operations where several tables have to be joined. On the other hand, a major advantage of SQL is its nearly universal use as a database query language.

- The XML application defined for OGC Web Feature Services [OCG02b] does not allow to use all abstractions in the conceptual model. Particularly, there is no support to use feature operations to compute new values (see [OCG02b], Section 1 and Section 7.3).

Results Language

The results of a data request to the mediator layer must be represented in a language to be used by other components. As with the query language, the language for results must be defined after the conceptual model because all its abstractions must be supported. Similarly to the query language, there are two approaches for this language: document-oriented, text-based representation languages, and API-oriented representation languages.

- *Document-oriented languages.* Languages of this type represent the complete results of a query using a text-based language. An example of this kind of languages is the Geography Markup Language (GML) [OCG03a]. The major advantage of these languages is that the query results can be used in any computing platform because there is no need for direct interaction with the software component that produces the document. On the other hand, these languages tend to be verbose and data manipulation is far from being efficient because the query results must be fetched as a whole and must also be manipulated as such.

- *API-oriented languages.* These languages are in some sense complementary to document-oriented languages. With these languages, query results are retrieved and manipulated using a programming interface offered by the software module that computes the results. The advantage of this approach is that the data producer can provide efficient data access by implementing data-caching, data buffering, and stream-based data access. On the other hand, this kind of languages requires that the interaction between the data producer and the data consumer must be technologically feasible. Examples of these languages are those defined for OLE/COM [OGC99a] and CORBA [OGC98].

The advantages and drawbacks of each kind of language are similar to those of query languages. Document-oriented languages are user friendly, but they need an API-wrapper to be used in computer languages. API-oriented languages are more difficult to use, but they are the only alternative in a programming language. Just as with query languages, both types are necessary in the architecture.

Metadata and Data Cataloguing

The importance of metadata and data cataloguing facilities has already been described in Section 2.7. In order to enable reusability of the information offered by the data tier, it is necessary to provide facilities for the description of the information. With respect to these facilities, the mediator layer must provide data abstractions for metadata and data cataloguing within the conceptual model, facilities for metadata and data cataloguing retrieval and manipulation within the query language, and the possibility to represent metadata and data cataloguing information within the results language.

4.4.2 The Data Sources Layer

The mediator tier provides a single conceptual model for the data tier that must be defined in a technology-independent way. However, the data tier must support multiple data sources of different types (e.g., data stored in files of the file system, a relational DBMS, or an extensible object-relational DBMS). These systems have different functionality and data is managed in very different ways in each of them. The data source layer is used to implement a mediator-wrapper architecture that allows the system to access data in many different types of data sources.

For each different data source type to be supported by the architecture, a *wrapper* module must be implemented. The topmost interface of these modules is similar to the interface offered by the mediator layer, that is, it must support the conceptual model, the query language, the results language and the metadata and data cataloguing facilities. This enables the mediator layer to deal similarly with any data source. Internally, each wrapper module must perform the conversions and processing necessary to perform these request against the particular data source. Thus, the wrapper module hides the peculiarities of a specific data source from the mediator layer.

Given that it may not be possible to implement all the functionality of the mediator layer for a particular data source type, it must be possible for wrapper modules not to implement the complete functionality. Therefore, it is necessary that the wrapper module provides metadata describing exactly which parts of the mediator layer are implemented, and which parts are unavailable.

Figure 4.4 shows four different wrapper modules: a *CAD-file wrapper*, an *extensible DBMS wrapper*, a *relational DBMS wrapper*, and an *OLE/COM wrapper*. Obviously, the number of different data sources is not limited to these four. We have chosen these for illustrative purposes.

The cad-file wrapper provides access to legacy CAD files (see Section 2.6.1). Such a wrapper module must implement most of the functionality of the conceptual module, which results in a very inefficient access to geographic information. However, this kind of wrapper must be provided at least to allow a way to import legacy data into more efficient data sources.

On the other hand, the extensible DBMS wrapper provides access to an extensible DBMS with a module for the management of geographic information (see Section 2.6.1). This wrapper is very simple because most of the functionality required is implemented in the DBMS extension module. The wrapper module must only deal with connection issues or conceptual model adaptation problems.

If a relational DBMS is used to store geographic information, the wrapper module uses tables or large objects to store geographic information and implements the query language using memory operations (see Section 2.6.1).

Finally, geographic information can also be accessible through a running application implementing a data service, for instance, a data service implementing the OGC specification for OLE/COM [OGC99a]. For this case, it is necessary to develop a wrapper module that deals with the connection, request and results transformation issues.

The preferred approach for data management in a GIS application that requires medium or advanced data manipulation uses the extensible DBMS wrapper because it is the most efficient. However, if only few data manipulation operations are needed, the overload in terms of storage space and computing power required by the extensible DBMS approach may be problematic. Instead, in this case it may be interesting to use a simple relational DBMS or files in the file system in order to achieve a light-weight architecture.

4.4.3 The Data Services Layer

As explained in Section 4.4.1, the mediator layer provides a conceptual model for geographic information, a query language, a language for representing results, and facilities for metadata and data cataloguing. However, different applications may require different interaction mechanisms with these facilities. For instance, a web-based GIS application requires an HTTP-based protocol for data requests, whereas a desktop application requires a programming language API. Similarly, some applications need all the functionality of the conceptual model, whereas other applications need only a subset of the functionality.

In order to build a flexible architecture, all these application profiles must be supported by the framework. Therefore, we propose to implement the *data services* layer on top of the mediator layer, to define specific profiles of the conceptual model for specific applications (i.e., subsets of the conceptual model, query and results language, and metadata facilities).

Two examples of services that can be implemented in this layer are the Web Feature Service and the Web Coverage Service defined by the OGC and described in Section 3.5. These specifications define a simpler way to access the conceptual model of the data tier, and therefore, must be implemented on top of the mediator layer.

4.5 Components of the Application Logic Tier

The application logic tier implements the business logic of each specific GIS application. In order to build a system architecture flexible and reusable, we propose that the functionality of this tier must be implemented by a collection of independent *services*, each responsible to perform a simple and well-defined task. When a task that requires the interaction of multiple services has to be performed, the services can be chained by means of a *workflow service*, which allows developers and end-users to build a new service by connecting a collection of simple services. The result is a highly-distributed architecture that ensures flexibility and reusability of the services. Since the services are not tied to a specific client, they can be used by all applications and can be moved to different locations according to the specific requirements of response time or other requirements. Figure 4.5 shows a detailed view of the architecture of the application logic tier.

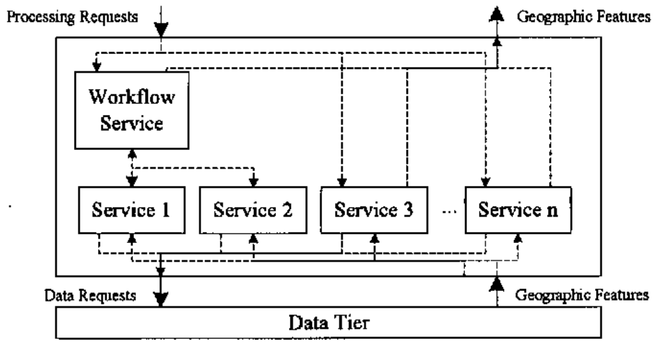


Figure 4.5: Architecture of the Application Logic Tier.

4.5.1 Services for GIS Applications

Each service must be defined by providing its interface, that is, the collection of operations that are implemented by the service, and the expected results after the invocation of each operation. When a service operation is invoked, the service module answers the request either by using its internal information, or by building and issuing the appropriate queries to the data tier and manipulating the data returned.

The services of a generic architecture for GIS application cannot be predefined because all the functionality necessary for GIS applications cannot be known in advance. Nevertheless, the ISO/TC 211 has proposed an extensive list of services in [ISO02j]. Now, we enumerate and describe briefly some of these services:

- *Coordinate Transformation.* This service can be used to change coordinates from one coordinate reference system to a second coordinate reference system. The service receives as input a feature collection, and produces as results the same feature collection referenced to a difference coordinate reference system.
- *Simplification of Geographic Values.* This service can be used to simplify geographic information in order to reduce the size of geographic data sets. The user provides references to the geographic data sets and the parameters for the simplification (i.e., method to be used and parameters for the method).
- *Route-finding.* Given a network defined by the user using the geographic information in the application schema, this service can be used to find routes from locations on the network, or to compute all the elements of a type that are reachable from a location.

Using the analysis that was presented in Chapter 2, and the experience gained after the development of the applications presented in Chapter 5, we propose the following services that are very interesting for the architecture of GIS applications.

- *Application Schema Transformation.* Different applications with different application schemas may need to share geographic information. Therefore, there is a need for a service that allows the user to define both application schemas and a mapping from one to the other to enable each application to access the information in the other application schema. Using this service and the mediator-wrapper architecture in the data tier, data transformation is possible between any type of data sources.

- *Information Validation.* There is a need for a service that allows the user to define validation rules (both alphanumeric and geographic), and ensures that the information inserted in the data tier fulfills these rules.

- *Automatic generalization service.* The problem of cartographic generalization has already been described in Section 2.5.1. This service provides the functionality needed to perform automatic generalization of geographic information. The user defines data structures defining the way in which generalization must be performed, and the service works as a filter over the conceptual model, providing geographic information that is automatically generalized.

The detailed description of standards for these services is outside the scope of our work. This is a task for organizations such as the OGC or the ISO/TC 211. Regardless of the new services that are defined, the architecture provides a place for them to be included.

4.5.2 Service Chaining

The services defined in the application logic tier must perform simple tasks in order to enable reusability and flexibility. However, when more complex tasks have to be performed, it is necessary to combine more than one service in a pipeline of services. This is often referred as *service chaining*.

The ISO/TC 211 has defined in [ISO02j] three different types of service chaining, according to the interaction mechanisms between the client of the service chain and the individual services that form the chain:

- *Transparent Chaining.* This type of service chaining is also called *client-controlled chaining*, because the client of the chain must control the complete process. In this type of chaining, the client is responsible for finding the services, performing the appropriate operation invocations, and dealing with the results or exceptions. This is the weakest type of chaining because it forces the client of the chain to control each detail of the process.

- *Opaque Chaining.* It is also referred to as *aggregate service.* In this type of chaining, instead of forcing the client to control the chain, a new service is defined to create and control the service chain. The service chain is also *static* in the sense that it is defined when the aggregate service is created and cannot be changed. Therefore, the service chain is completely opaque to the client because it does not know whether the process is performed by a simple or an aggregate service. However, a drawback of this type of chaining is that all flexibility is lost because the client cannot control the process.

- *Translucent Chaining.* This type of service chaining is an intermediate alternative to the previous ones. Here, a generic *workflow service* is defined that is used by the client to create and control the service chain. The client describes the service chain using a high-level modeling language such as the one sketched by the ISO/TC 211 in [ISO02j], and specifies different parameters that control the process (e.g., service timeouts, or exception handling procedures). Then, the workflow service creates and controls the chain using the information provided by the client.

  This type of service chaining has the advantages of both the transparent and the opaque chaining. The client has some control over the process but does not have to deal with each detail. However, the drawback of this type of chaining is the complexity of designing and developing a workflow service.

In order to be able to implement the concept of service chaining in the application logic tier, two additional issues must be taken into account:

- *Service Metadata.* Not only geographic information must be described in order to allow end-users to understand its semantics. Metadata must also be provided for services to be enable the user to understand the services in order to create service chains. This service metadata must include a detailed description of the service (e.g., available operations, signatures of the operations, accepted formats, or error exceptions).

- *Parameter Passing Procedures.* Once the service chain is created, information is passed from one service to the other using operation arguments. It is important that services are defined in such a way that the arguments of an operation can be fetched from another service in addition to being passed in the invocation.

  For instance, the operations of the Web Map Service defined by the OGC [OCG02c] can fetch geographic features to be portrayed from any of the OGC data services [OCG02b, OCG03b]. Due to the web-oriented nature of these services, they exchange arguments as a complete unit. That is, a service in the chain must wait until all arguments have been completely received before starting the execution of the operation. This is a very inefficient approach. Instead, where possible, a *stream-based* solution must be designed that acts as a real pipeline executing all the operations in the chain in parallel.

4.6 Components of the Presentation Tier

The presentation tier implements the user interface of the application, which enables data visualization, data manipulation and data entry. This tier receives all user interaction in the form of mouse gestures, keyboard inputs or inputs from any other device, evaluates it, and generates the appropriate response using either functionality within the presentation tier, or invoking operations of the application logic tier or the data tier. After the response is generated, the results are displayed to the user using the appropriate user interface controls and visualization metaphors. We propose that the functionality of this tier must be divided into three layers: the *client functionality* layer, the *map display* layer and the *portrayal* layer. The portrayal layer converts the geographic features returned by the application logic tier into cartographic objects, which are then rendered by the map display layer using a graphical format. Finally, the client functionality layer displays the map along with any other user interface component, and deals with user interaction. Figure 4.6 shows a detailed description of the architecture of the tier. In the following sections, we describe each layer independently.

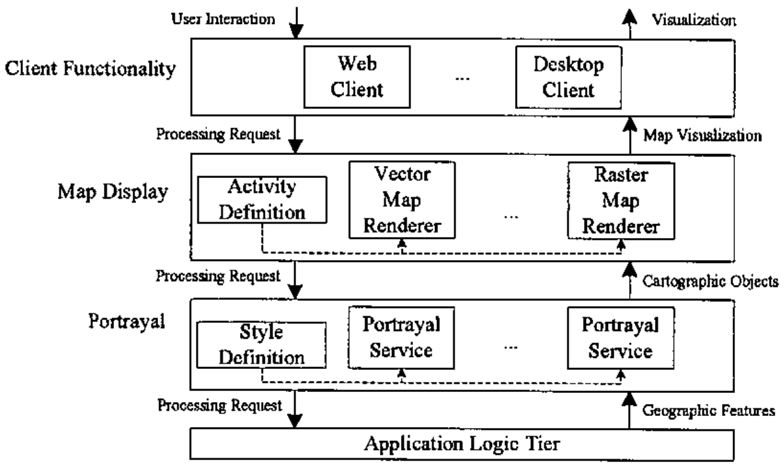


Figure 4.6: Architecture of the Presentation Tier.

4.6.1 Portrayal Layer

As described in Section 2.5.1, the abstractions used to visualize geographic information are different from those used to represent them in the computer. Therefore, the first process that must be carried out by the presentation tier is converting the collection of geographic features returned by the application logic tier into a collection of cartographic objects in a presentation-oriented data format (e.g., SVG, JPG).

There are many different data formats suitable for map presentation, which can be divided into two categories:

- *Raster image format.* The image is represented using a grid of pixels. Some examples of raster image formats are PNG or JPEG.
- *Vector image format.* A collection of geometric shapes is used to represent the image. Some examples of these formats are CGM and SVG.

Given that there are many different graphical formats, each with advantages over the others, and many different display devices (e.g., computer devices, or portable devices such as mobile phones or PDAs), we propose that the portrayal layer must be composed of a collection of different portrayal services with a similar interface but producing different output formats. For instance, a portrayal service can produce maps in an image format such as PNG or JPEG, where the resulting images are small and therefore suitable for transmission over low-bandwidth data networks. On the other hand, another portrayal service can produce maps represented in a vector image format such as SVG or CGM, which results in larger images that can be rendered at a higher-quality but require high-speed data networks. Moreover, another portrayal service can produce printable maps using the Adobe PDF format. These maps can be stored on the client computer and be printed with high quality when needed. This portrayal service can also automatically produce *map series* using a single page template to create a collection of pages for contiguous areas.

The basic abstraction of the portrayal service is the *cartographic layer*. A cartographic layer is a collection of semantically-related cartographic objects that are displayed as a unit in the map. This is the basic unit that can be defined and be managed by the user. The portrayal process of a cartographic layer is specified by a *style definition*, which consists of:

- *The specification of the geographic features that are displayed.* For each cartographic layer, the user must specify the data source and data abstractions used to fetch the geographic features. This is often expressed as a set of processing requests for the application logic tier or data requests for the data tier.

- *The set of map and data properties that must hold for the geographic features to be visible in the map.* In addition to specifying the data to be portrayed, the user defines the conditions that must hold for each geographic feature to be visible. This enables the user to define maps where the geographic features are displayed using different colors according to the value of an attribute by assigning different graphical styles to the geographic features with respect to the result of a query.
- *The style to be applied to the geographic features to create the cartographic objects.* Finally, the user must also specify the graphical style of the cartographic objects. This is done by assigning stroke color and styles for the lines, fill color and styles for the surfaces, label presentation styles for the texts, and symbols for the points.

It can be seen that a cartographic layer is an abstraction that provides a mapping from geographic features to cartographic objects in such a way that the end-user manipulates only one abstraction at the presentation tier while the representation in the map may vary in style, resolution or other aspects. Consider, for instance, the following example expressed in an *ad hoc* language only for illustrative purposes:

```
create roads as cartographic layer {
  if map-scale < 1:10000 then
    display roads.surface using red-surface-style
  if map-scale >= 1:10000 and map-scale < 1:100000 then
    display roads.line using red-line-style
  if map-scale >= 1:100000 then
    display simplify(roads.line) using red-line-style
}
```

In this example, the user defines the cartographic layer `roads`. The geographic features of this cartographic layer are produced in three different ways according to the value of one property of the map (i.e., the map scale). If the map scale is large, the surface of the road is displayed in the map using a graphical style defined elsewhere (i.e., *red-surface-style*). The geographic value is request to the data tier directly (represented by `roads.surface`). For intermediate map scales, the road is displayed as a line using a request to the data tier that fetches a different geographic attribute of the geographic feature `roads` (i.e., the request `roads.line`). Finally, for small scale maps, the road is represented as a simplified line using a request to the application logic tier that uses a simplification service to reduce the detail of the geographic values (i.e., the request `simplify(roads.line)`).

A portrayal service defined and implemented along these lines handles multiple cartographic representations for an object transparently, thus enabling cartographic generalization (see Section 2.5.1). This is possible because an object at the presentation tier (e.g., a road section) can be associated to multiple objects in the data tier with

Event name	Description
mouse in	Occurs when the mouse cursor is moved into a cartographic object
mouse over	Occurs whenever the mouse cursor is moved inside a cartographic object
mouse out	Occurs when the mouse cursor is moved out of a cartographic object
mouse click	Occurs when the mouse button is clicked over a cartographic object
map pan	Occurs when a movement of the map view is finished
map zoom	Occurs when the change of map scale is finished
map load	Occurs when the loading of the map is finished

Table 4.1: Events of the Activity Language.

different resolutions and representations (e.g., a line for small map scales and a surface for large map scales). The style definition allows the user to specify different sets of geographic features that must be displayed for a particular combination of map and data properties using a specific graphical style.

There already exists a mature specification for a map portrayal service, the SLD-enabled Web Map Service specification [OCG02c, OCG02a] defined by the OGC. Our architecture provides a software layer where this specification can be easily integrated.

4.6.2 Map Display Layer

The map display layer is in charge of visualizing the cartographic objects computed by the portrayal layer using a *map display* component. This component uses a *map metaphor* that defines the ways in which the end-users can visualize and manipulate the displayed map. For instance, the map scale can be changed using the *zoom-in* and *zoom-out* operations of the metaphor, and the information displayed in the map can be changed using the operations *add-layer* and *remove-layer* (see Section 2.5.1).

Moreover, the map display layer is also in charge of providing the functionality to manipulate the map, as well as invoking the operations of the application logic tier. The developer associates actions to user interface events that occur in the map. These actions are implemented by the developer using the operations available in the application logic tier and the data tier. Table 4.1 illustrates the type of events that can be recognized by the map display component.

As an example, a developer can associate the event *click over an element in the map* to the action *display element information*. The list of events is obviously not exhaustive. Moreover, a language to represent the association of actions to events must be designed. There is currently no specification regarding this language, but as a result of our implementing work for the applications described in Chapter 5, we propose that a standard specification must be defined for such a language.

The image format for the map selected in the portrayal service determines the functionality available at the map display component. For instance, individual cartographic objects cannot be easily identified in a raster image format, thus reducing the possible events that can be recognized by the map display component. On the other hand, vector image formats use geometric shapes to represent cartographic objects, and therefore can identify separate objects and recognize more events.

4.6.3 Client Functionality Layer

The client functionality layer implements the user interface providing the graphical controls that the end-user visualizes and manipulates. The layer is composed of modules that implement different *views* of the user interface. A view consists of a particular combination of user interface components and interaction mechanisms adapted for a particular computing platform. For instance, Figure 4.6 shows two different views in the client functionality layer: a *web-based* client that implements the user interface taking into consideration the peculiarities of web technology, and a *desktop* client implemented with a traditional procedural programming language for a desktop-based GIS application.

If the whole of the functionality of the system were implemented in the application logic tier, the response time of the system would not be fast enough. Each user interaction would require a request to the application logic tier, which would in turn require a request to the data tier. This implies a strong communication overhead that would result in reduced efficiency of the application. Instead, and in order to achieve faster response times from the system, the presentation tier may implement some basic functionality for simple computations. For instance:

- *Client-side zoom and pan of the map view.* When the user invokes a pan or a zoom operation on the map display component, the new map is first computed using the information in the map rendered by the map display layer while a new map is simultaneously requested. This helps improving the response time because the application does not stop waiting for the new map to be transmitted from the server. Instead, a temporary map is displayed right away, and the new map is fetched in the background.

- *Measure distances and areas.* Simple area and distance measurements can be performed at this layer without the need of performing a request to the application logic tier.

- *Management of the graphical legend.* Each map displayed on the screen must be accompanied by a graphical legend that specifies real world phenomena that are described by each graphical symbol in the map. The management of this graphical legend is a task that can be accomplished at this layer.

- *Display the map context.* In order to facilitate the use of the map, it is important to display context information of the map that is being displayed. For instance, the display of an overview map, a graphical scale bar, or a north arrow can be managed by the client functionality layer.

4.7 Building Specific Applications

We have described in the previous sections the components of a generic architecture for GIS applications. It is clear from this description that the functionality implemented by many of the components in the architecture is independent of any particular GIS application. Therefore it is possible to implement the components in a generic way and adapt them later with application-specific details. The following generic services and components can be developed:

- *Data Source Wrappers.* Each data source wrapper can be implemented independently of the GIS application (e.g., JDBC or ODBC drivers). The connection information and the application schema can be given using a high-level language.
- *Mediator Layer.* This layer can be implemented independently of the application schema. For each particular application, the application schema and the metadata information can be given using a formal modeling language (e.g., UML).
- *Data Services.* Generic data services can be developed. The application schema information is described using a formal language (e.g., the OGC specifications for the WFS [OCG02b] and the WCS [OCG03b]).
- *Services of the Application Logic Tier.* The processing services and the workflow service are independent of the actual application schema.
- *Portrayal Services.* Each portrayal service is independent of the application schema. The application-specific details are given as style definitions expressed in a high-level language.
- *Map Display Components.* Specific map-display components for particular computer platforms can be developed independently from the application. The information regarding the portrayal service to be used and the activity rules is given using a high-level language.
- *User-Interface Clients.* Each individual user interface view can be implemented independently of the application schema, which is described using a high-level language.

In order to build an specific application, the following steps must be followed:

- *Design the architecture.* Using the requirements of the application as a guideline, the appropriate services are chosen for each layer in the architecture.
- *Configure the services.* Each service must be configured with the application-specific information, such as the application schema, the service chains that must be configured and built, or the style definitions.
- *Tune the architecture.* Additional application-specific functionality may have to be implemented as additional modules of the architecture using computer programming languages.

Using metadata for the components of the architecture, this process can partially be automated for specific GIS applications. An application that allows a developer to configure the system architecture using visual interaction mechanisms can be built. Moreover, visual editors can be provided to create and manipulate the configuration languages of each service.

4.8 Summary

Considering the quality of the work developed by the OGC and the ISO/TC 211, it is clear that the architecture that we have proposed must be (and is in fact) strongly inspired by the specifications published by these organizations. This implies that the three tiers of the architecture are similar in purpose to those defined by the OGC, including the organization of the tasks and the strict separation of the modules that interact only using well-defined interfaces. Furthermore, we have used the standards adopted by the OGC and the ISO wherever it is possible.

However, our proposal extends these specifications in many senses:

- *Additional detail.* We have provided additional detail in the definition of the software tier of the architecture. Particularly, we have proposed a mediator-wrapper architecture for the data tier, and a data services layer to define specific application-profiles of the mediator layer. We have also proposed a separation of the functionality in the presentation tier into three independent software layers.
- *Additional services.* There are no specifications proposed for some of the services and components that we have described. For instance, we have proposed a map display client that deals with events at the user interface.

The main contribution of this chapter is the definition of an architecture that incorporates the standard specifications defined by the OGC and the ISO/TC 211, and proposes new components where there are no standards proposed [LBPV04a, LBPV04b]. We have not defined a complete and precise specification of these new components because this is a task for organizations like the OGC and the ISO/TC 211.

Furthermore, the architecture provides a framework for the integration of future standards developed for data representation, manipulation, and visualization. Finally, we have made explicit in our architecture some concepts that are implicit in the specifications.

Chapter 5

The EIEL Project: An Example of the Application of the Architecture

5.1 Introduction

In order to prove the applicability of our proposal, we developed a GIS application using the system architecture and the concepts that we describe in Chapter 4. By using a real world problem to test these concepts, we ensure that we face actual problems that can be found in the development of GIS applications.

The development of the GIS applications for the EIEL project required five people working during three years to be completed. In addition, we used commercial GIS development tools instead of developing custom-made software modules. This allowed us to reduce the costs and the development time that custom-made tools imply. These applications, and the analysis of the problems found in the application of the architecture for their development is the third contribution of our work.

In this chapter, we describe our experience in the development of this GIS application. First, we give in Section 5.2 important background information on the project of which the GIS application is a part. Then, in Section 5.3, we describe the application schema of the GIS in more detail to give a hint of the vast amount of information that must be managed by the developed GIS. The chapter follows with a description of the applications that were developed for the GIS. First, we give in Section 5.4 an overview of the data workflow and a brief description of the applications. Then, the web-based application for the exploitation of the information stored in the database is described in more detail in Section 5.5 because it shows a direct application of the architecture and concepts described in our proposal. Finally, we summarize the chapter in Section 5.6.

5.2 The EIEL Project

In order to optimize the management of the country, the Spanish territory and the government responsibilities are divided into a hierarchy of four levels. The central government deals with issues concerning the entire country, such as foreign affairs. The country is 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 Council of each province 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 each Provincial Council that results in a database with standardized and homogeneous information in terms of the application schema, geographic scope, survey time, data formats, and evaluation criteria. This information can be used to compute indicators and to 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. This database is composed of 133 relations with an average of 20 attributes per relation. The greatest number of attributes in a relation is 100 and the least number is 10.

The province of A Coruña is located in northwestern Spain. With more than one million inhabitants and almost eight thousand square kilometers, it is densely populated with more than a hundred and twenty-five inhabitants per square kilometer. The Provincial Council of A Coruña decided to broaden the goals of the EIEL for the year 2000. More 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 a geographic information system with the information collected. This system is going to be used by the Provincial Council staff. Moreover, a publicly-accessible, web-based application enables the inhabitants of the province (actually, everybody connected to the Internet) to browse the information collected.

These goals were 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 laboratory 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 [BCF$^+$03].

The result of this project is a web-based GIS application that is a powerful analysis tool in which many different evaluation mechanisms can be used and many *indicators* were produced to analyze the living conditions in the province. The existence of geographic information attached to the alphanumeric information allows end-users to perform geographical analysis with 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 task.

Before presenting a detailed description of the application schema, we can give some quantitative information about the database. Information was collected for the 94 municipalities in the province, ranging from general information about the municipality (e.g., population) to detailed information of each village. The database contains information about 4000 towns and villages, more than 35000 road sections, more than 40000 street sections, approximately 17000 water supply pipe sections and 17000 sewage treatment pipe sections. Moreover, information about services in the municipality was collected. This includes information about 700 education centers, 1200 sport facilities, 1000 parks and gardens, 800 culture facilities, and 200 health centers. The geographic location and/or extent of more that 100 different kinds of element was surveyed, including municipality borders, village built-up areas, road section surfaces, pipe section paths, or buildings. The database totals around three gigabytes of information.

5.3 The EIEL Application Schema

We have used the Unified Modeling Language (UML) to describe the EIEL application schema. For the sake of clarity, only the geographic attributes of the classes in the model are depicted. The information represented by each class is described in the text. The data type of each geographic attribute is given using a symbol

The main goal of the project was to build a database that stores information regarding the infrastructure and equipment in the province, and the municipalities that are served by each element. However, some of the elements that render services in the province do not belong to the province. Border municipalities may receive services from elements placed in the neighboring provinces (e.g., hospitals, water sources, or population centers). Even though it is not necessary to collect detailed information of these elements (they are surveyed by the Provincial Council in the neighboring municipality), it is necessary to represent at least that a service is rendered by an element that is outside the border of the province. This is represented in the application schema by using one class to identify the elements, and a specialization of this class to represent the elements that are actually surveyed.

As an example, consider the case of water sources. Not all water sources are geographically located in the province. Therefore, there is a class named *Water Source* that represents all water sources, regardless of their location. The attributes of this class only represent the identifier of the water source. A specialization of this class, named *Surveyed Water Source*, is used to represent the data collected for each water source. One advantage of applying this concept in the design of the application schema is that it is possible to define rules to validate the data without having to deal with special cases. Considering the same example, it is possible to define a validation rule such as:

There must be at least one water source for each population center that has water distribution network.

If only surveyed elements were represented by the application schema, then all population centers whose water source were not located in the province would have to be treated as special cases.

Another concept applied to many classes in the application schema is that two different representations are stored for some geographical elements, e.g. road sections, population centers, or power plants. Sometimes this is done to be able to use a different representation to solve different problems. For instance, in order to apply graph algorithms to the road network, a line-based representation is more appropriate. To compute statistics (e.g., area or average width) or make decisions (e.g., to locate places where the road gets narrow), the surface is needed. However, on other entities, this duplicity of geographic information is used to implement a multi-scale, multiple representation visualization system.

Since the application schema is very large, we have divided it into smaller diagrams that share some key classes. The central classes of the application schema (i.e., those that are present in all parts) are those that describe the territorial structure of the province, which are described in Section 5.3.1 Then, we describe in Section 5.3.2 the part of the application schema that deals with the road network. Waste disposal and the street lighting system are described in Section 5.3.3. Then, the water distribution system, the water equipment elements and the sewage disposal system are described in Section 5.3.4 and Section 5.3.5. The application schema that represents the equipment elements in each municipality is described in Section 5.3.6. The last part of the application schema, which we describe in Section 5.3.7, includes the classes that represent the geographic information that is used as context for the maps.

5.3.1 Territorial Information

Figure 5.2 shows the classes and relationships of the application schema that represent the administrative division of the province, the urban planning and the collaboration relationships between the municipalities. These classes are the center of the application schema because they appear in all subschemas. The symbols used to represent the data type of geographic attributes are shown in Figure 5.1. The meaning of each data type is described in Table 5.1.

The class *Province* represents each province of Spain, and stores the boundary of the province in the geographic attribute *border*. Each province consists of a number of municipalities that are represented by the class *Municipality*. Only the municipalities belonging to the province of A Coruña are surveyed by the Provincial Council of A Coruña. These municipalities are represented by the class *Surveyed Municipality*. Each municipality consists of a number of population entities, which are a partition

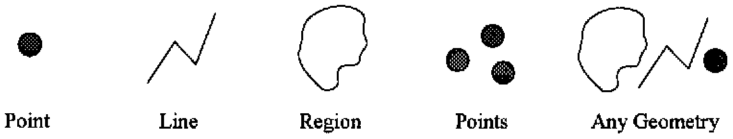


Figure 5.1: Geometry Types.

Data Type Name	Meaning
Point	A single point in the Euclidean plane. Used to store the position of an object
Line	A collection of line segments, used to describe a network
Region	A collection of polygons with (optional) holes, used to represent surfaces
Points	A collection of point values, used to represent the position of a set of objects without information of each individual element
Any geometry	A collection of geometry values from any type

Table 5.1: Geographic Data Types Used In The Application Schema.

of the municipality into disjoint areas in order to facilitate management. Within each population entity, there is a number of population centers, which are represented by the class *Surveyed Population Center*. For each population center, general information regarding the living conditions in the village or town is collected. For example, the number of inhabitants, number of free and occupied familiar dwellings, number of hotel rooms, camping places, number of inhabitants with appropriate water supply and sewage disposal, number of inhabitants with deficit in water supply and sewage disposal, or radio and TV reception quality. Moreover, the boundary and the center of each population center is represented using the geographic attributes *border* and *center* respectively.

The descriptive attributes of the class *Surveyed Municipality* store the same information as the class *Surveyed Population Center* for the population of the municipality that do not live in a population center. For instance, the number of inhabitants in each population center is represented by an attribute in the class *Surveyed Population Center*, and the number of inhabitants in the municipality that do not live in a population center is represented by an attribute in the class *Surveyed Municipality*. Furthermore, the class *Surveyed Municipality* has a large number of geographic attributes that are described in Table 5.2. Only the location or extent of these elements is stored, no further descriptive information is collected.

The urban planning of each municipality is also represented in the application schema. The areas of the municipality dedicated to each particular usage are represent as geographic attributes of the class *Urban Plan*, which also stores descriptive

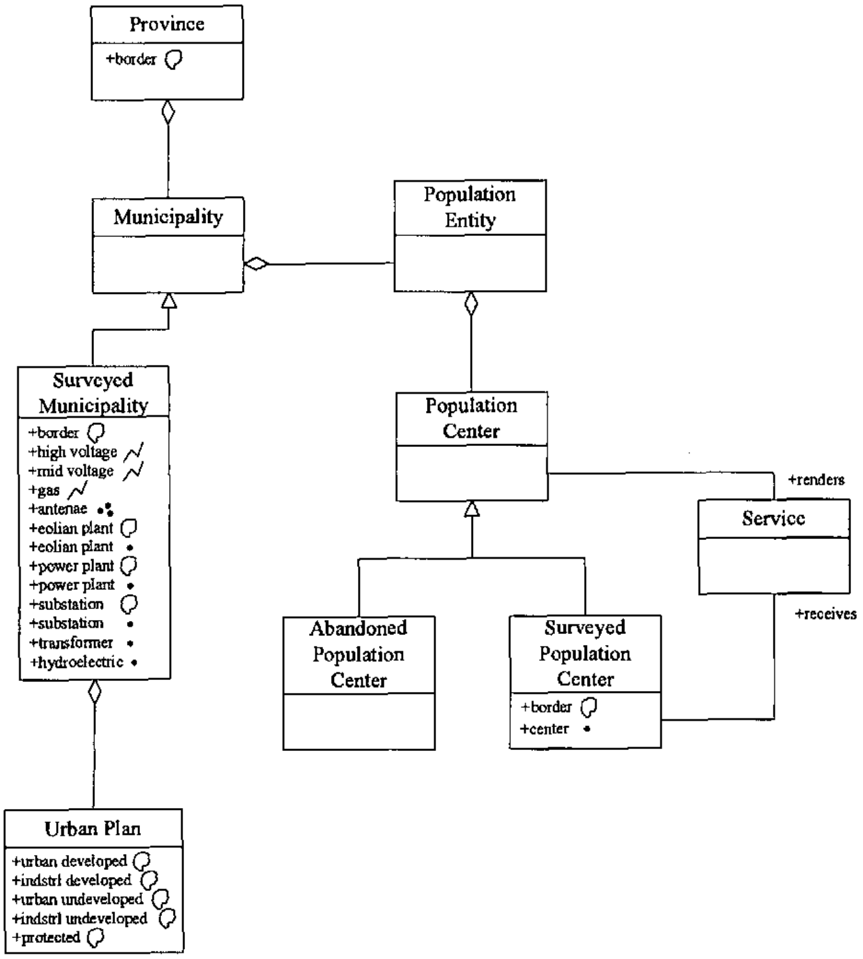


Figure 5.2: Territory.

Attribute Name	Meaning
border	Municipality boundary
high voltage	High voltage electricity network
mid voltage	Medium voltage electricity network
gas	Gas distribution network
antenae	Location of communication antenae
eolian plant	Location and extent of eolian power plants
hydroelectric	Location of hydroelectric power plants
power plant	Location and extent of other power plants
substation	Location of electricity network substations
transformer	Location of electricity network transforming stations

Table 5.2: Geographic Attributes of the Class *Surveyed Municipality*.

Attribute Name	Meaning
urban developed	Urban area in the municipality already developed
indstrl developed	Industrial area in the municipality already developed
urban undeveloped	Urban area in the municipality yet to be developed
indstrl undeveloped	Industrial area in the municipality yet to be developed
protected	Area in the municipality protected due to its ecological value

Table 5.3: Geographic Attributes of the Class *Urban Plan*.

information about the plan itself (e.g., approval date). The meaning of each geographic attribute is given by Table 5.3.

Finally, not all services can be provided at each population center. Sometimes the inhabitants of a population center must move to another population center to receive a particular service such as education, or health care. This relationship between population centers is represented by the class *Service*.

5.3.2 Road Network

Figure 5.3 shows the application schema for the road and street network. It includes the classes that represent the territorial division of the province in municipalities and population centers, and classes that represent roads, streets and sidewalks.

The roads of the province are represented by the class *Road*. This class represents the complete road from its begin to its end, and stores general information of the road such as the route number. The road route is divided into sections that have the same characteristics in terms of the values of the following descriptive attributes:

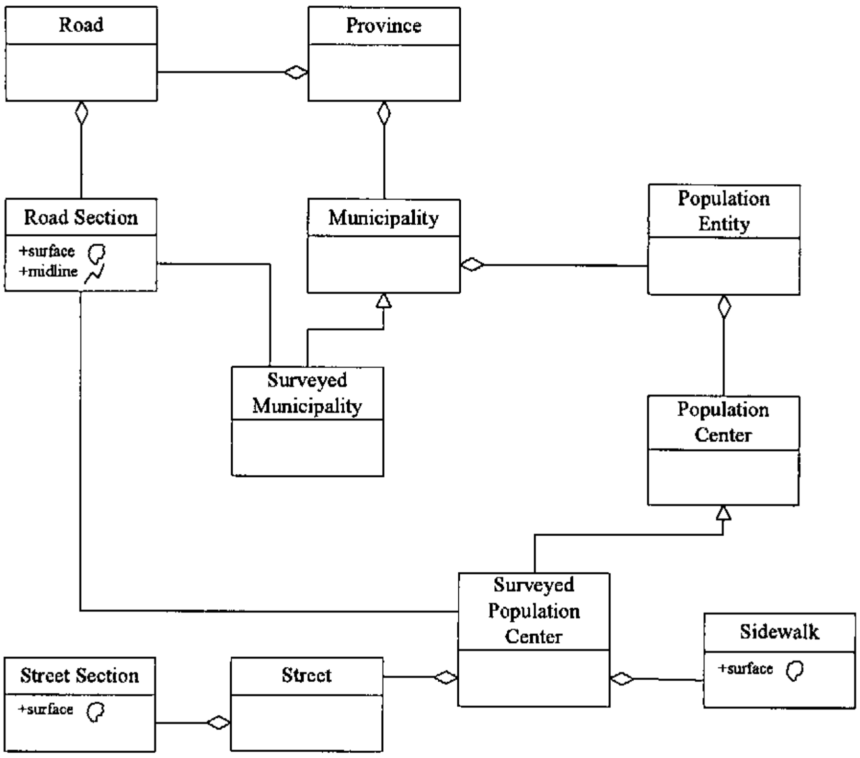


Figure 5.3: Road Network.

- *Organization that owns the road section.*
- *Organization that maintains the road section.*
- *Type and state of the road surface.*
- *State and adequacy of road signs.*
- *Width and its adequacy.*

Road sections are represented by the class *Road Section*. Furthermore, the surface and middle line of the road section are represented by the geographic attributes *surface* and *midline* respectively.

The street network is represented by the classes *Street* and *Street Section* in a similar manner. The general information of the streets in a surveyed population center is represented by the class *Street* (e.g., the street name). Then, the street surface is divided into sections with the same characteristics in terms of the following descriptive attributes:

- *Type and state of the surface.*
- *Width.*
- *Sidewalk presence.*

Street sections are represented by the class *Street Section*, and the geographic attribute *surface* represents the surface of the street section. Finally, street sidewalks are represented by the class *Sidewalk* and their surface is represented by the geographic attribute *surface*.

5.3.3 Waste Disposal and Street Lighting

The part of the application schema that represents waste management and street lighting is shown in Figure 5.4. The classes that represent the division of the province into municipalities and population centers are again included in the application schema. In addition to that, the schema includes classes that represent information regarding dumping sites, the waste collection service, and the street lighting of the municipality.

There is a number of dumping sites in each municipality of the province. Each surveyed dumping site is represented by the class *Surveyed Dumping Site*, and is described by means of attributes storing the type of dumping site, its problems, its capacity, and its expected life period. Moreover, each dumping site serves a population center, and its location and extent are represented by the geographic attributes *center* and *border* respectively. Waste collection information is represented by the class *Waste*

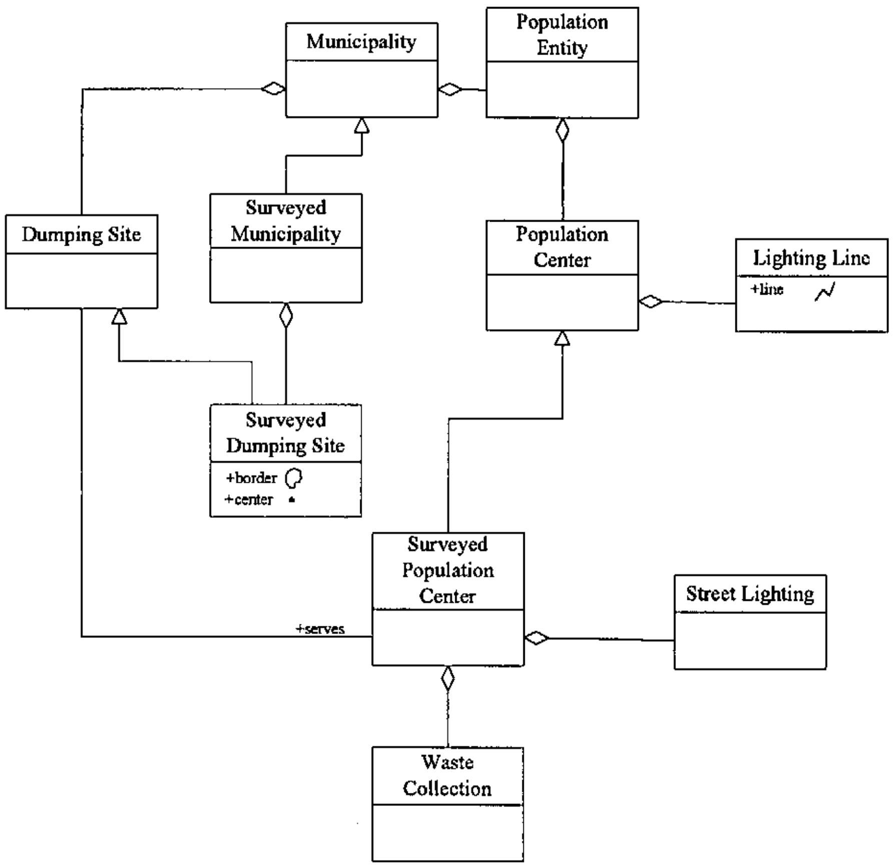


Figure 5.4: Waste Disposal and Street Lighting.

Collection for each surveyed population center. This information includes data such as waste production volume, or collection periodicity and adequacy.

On the other hand, street lighting information is represented by the class *Street Lighting* that is also associated to each surveyed population center. This class stores information such as the number of street lamps in the population center, the installed street lighting power, or the state of the street lighting. The application schema does not represent information for each individual street lamp, only the location of the street lamps is represented. This is carried out by the class *Lighting Line* and by the attribute *streetlamps* of the class *Municipality*. The former represents the street lamps located in a population center that are equally spaced. This is done by means of a line that connects all the street lamps and the distance between each pair of street lamps. The latter represents all other street lamps as a collection of points.

5.3.4 Water Distribution Network

Figure 5.5 shows the application schema for the water supply network from the water sources to the distribution network in the population centers. In addition to the classes that represent the municipalities and the population centers, the schema includes classes to represent water sources, raw water networks, water treatment processes, water tanks, distribution networks, network deficiencies and water springs.

The water distribution network is represented in the application schema using five different classes. These classes, enumerated from the water source to the distribution network that carries the water to the buildings, are the following: water sources, raw water network, water treatment, water tank and water distribution network. Each of these elements is associated to the municipality in which it is located, and to the population center that is served by the element.

A water source represents the location where water is drawn for the supply network. Each surveyed water source is represented by the class *Surveyed Water Source*, and the location is represented by the geographic attribute *location*. Moreover, the descriptive attributes of the class represent the type of the water source (e.g., a spring, a dam, or a well), the river basin, the organizations that own and manage the water source, the organization that manages the water source, and the state of the water source.

Water storage elements in the network are represented by the class *Surveyed Water Tank*. The location of the tank is represented by the geographic attribute *location*, and a detailed description of the water tank is given by the descriptive attributes. These attributes represent the position of the tank (i.e., whether it is buried, on the surface, or above ground level), the organizations that own and manage the water tank, the capacity and the state of the water tank.

The class *Surveyed Water Treatment* represents the water treatment processes performed on a water supply network, either manually or at a treatment plant. The

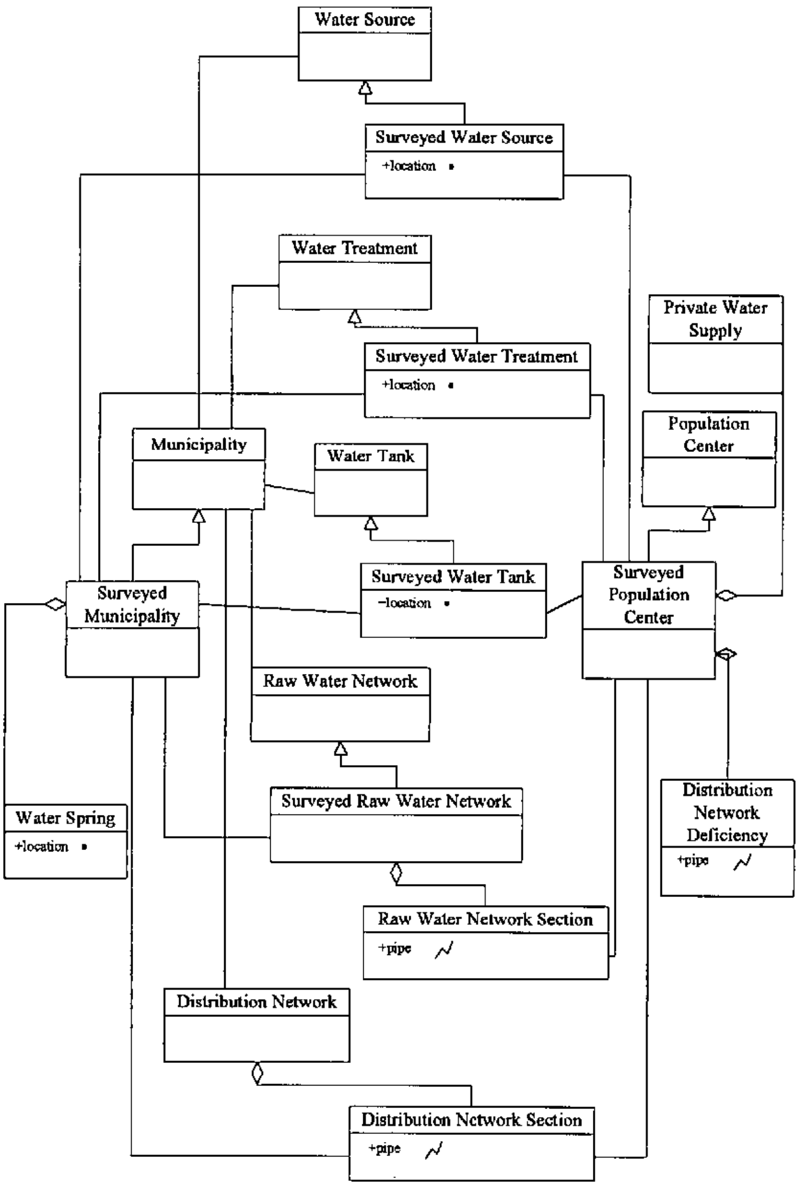


Figure 5.5: Water.

geographic attribute *location* represents the location where the treatment is performed. The descriptive attributes of the class describe exhaustively the type of treatment process that is performed on the water supply network, including quality controls, periodicity, or chemical components used.

The network that carries the water from the water source to the buildings is divided into two different kinds of network, the raw water network and the distribution network. Both kind of networks are further divided according to their connectivity, and each individual network of each kind is represented by an object of the class *Surveyed Raw Water Network* or the class *Distribution Network*. Then, each individual network consists of a set of connected pipe sections represented by the classes *Raw Water Network Section* and *Distribution Network Section*. The geographic attribute *pipe* represents the course of each pipe section, and the descriptive attributes represent the diameter, material, state, and organizations that own and manage the pipe section.

The location of the water springs that are not part of the distribution network is represented by the class *Water Spring*. The location of the water spring is represented by the geographic attribute *location*, and the information that represents precisely the type of water spring is represented by the descriptive attributes of the class.

The final element of this part of the application schema is the class *Distribution Network Deficiency*. It represents the pipe sections that are needed to solve the deficiency in the distribution network by means of a geographic attribute named *pipe*.

In addition to the elements shown in the application schema described in Figure 5.5, each distribution network is associated to the classes that can be seen in Figure 5.6. These classes represent the position of the irrigation hydrants, the fire hydrants, the cut valves, the vents, the drinking fountains, and the pumping stations. Each class represent the position of the element by means of the geographic attribute *location*.

5.3.5 Sewage Disposal

The application schema for the sewage disposal network is shown in Figure 5.7. This part of the application schema represents the network that carries the sewage from the building to the outfall. This network consists of the following elements: drainage network, sewer main, sewage treatment, and outlet network. Additionally, information regarding the drainage network deficiency and the outfalls is also represented.

Similarly to the application schema for the distribution network, all the elements of the sewage disposal network are associated to the municipality in which they are located, and to the population center that is served by the element.

The sewage network is divided into three kinds of networks, the drainage network, the sewer main and the outlet network. The first kind of network groups the pipe sections that have buildings connected and drain the sewage from them. The second kind of network consists of pipe section that do not have buildings directly connected,

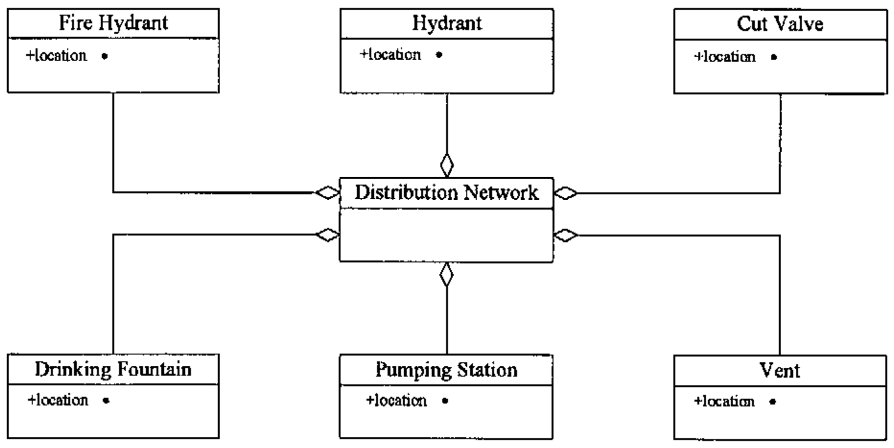


Figure 5.6: Water Equipment.

and conducts the sewage to the treatment plant. Finally, the third kind of network consist of pipe sections that carry the treated sewage to the outfall. Independently of the kind of network, pipe sections are grouped into networks according to connectivity. Each network is represented by a class (either *Surveyed Drainage Network*, *Surveyed Sewer Main*, or *Surveyed Outlet Network*), and each individual pipe section is represented by a class (either *Drainage Network Section*, *Sewer Main Section*, or *Outlet Network Section*). The course of each pipe is represented by the geographic attribute *pipe*, and the diameter, material, state, and organizations that own and manage the pipe section are represented by descriptive attributes of the class.

Sewage treatment processes, which can be performed manually or at a plant, are represented by the class *Surveyed Sewage Treatment*. This class represents the location where the treatment is performed using the geographic attribute *location*. Furthermore, a collection of descriptive attributes provide a throughout description of the process performed.

The class *outfall* represents the location and the properties of the place where the sewage is released in the environment. The location is represented by the geographic attribute *location*, and the distance from the outfall to the nearest population center and other data regarding the properties of the outfall are represented by descriptive attributes.

Finally, the pipe sections needed to solve the drainage network deficiency are represented by the class *Drainage Network Deficiency*. The course of the pipe section is represented by the geographic attribute *pipe*.

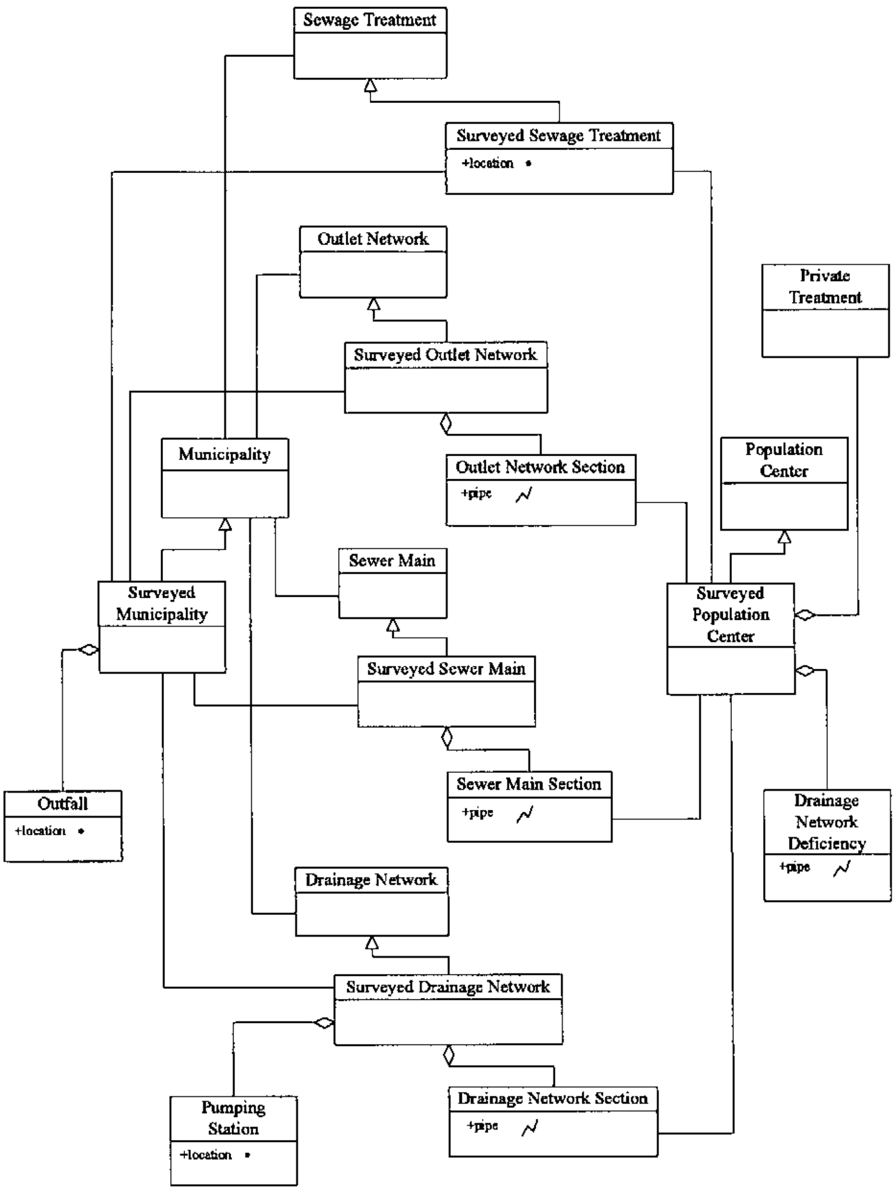


Figure 5.7: Sewerage.

5.3.6 Equipment

The application schema that represents the equipment of the municipality is shown in Figure 5.8. We use the term equipment to refer to the set of resources serving to equip a municipality in order to render a service to the population. Each equipment is associated to the population center where it is located. The different kinds of municipality equipment are: cemeteries, slaughterhouses, morgues, health centers, sports centers, marketplaces, cultural centers, welfare centers, education centers, parks, civil defense, churches, public buildings, unused public buildings, and hotels.

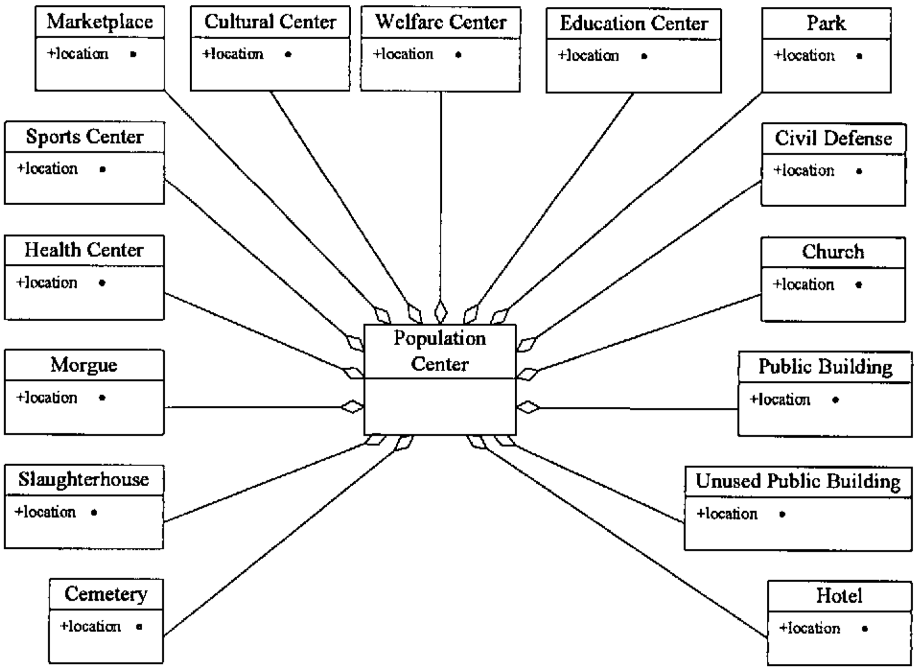


Figure 5.8: Equipment.

Each class represents the location and the land parcel of the equipment using the geographic attributes *location* and *parcel* respectively. Additionally, each class represents the following information using descriptive attributes: conservation state, organizations that own and manage the equipment, and the surface of the parcel that is located indoor and outdoor.

indoor, outdoor and parcel surface.

Furthermore, the following information is collected for each class using descriptive attribute:

- *Cemetery:* Distance to the nearest population center, suitability of accesses, presence of chapels and morgues, degree of saturation, and extension possibilities.
- *Slaughterhouse:* Type, capacity, percentage of usage, and type of livestock for which the equipment is suited.
- *Morgue:* Number of rooms.
- *Health center:* Type, number of beds, and presence of intensive care unit.
- *Sports center:* Type of sports played.
- *Marketplace:* Type of marketplace (e.g., fishmarket, or livestock market).
- *Cultural center:* Type of cultural center (e.g., library, museum, or theater).
- *Welfare center:* Type and maximum number of occupants.
- *Education center:* Levels taught, pupils per level, and number of classrooms.
- *Park:* Presence of water, electricity, sanitary fittings, or playground areas. This class also represents gardens and other nature-related areas.
- *Civil defense:* Type, scope, number of professional and volunteer members, and amount and type of rescue equipment.
- *Public building:* Usage type (e.g., city council, warehouse, or post office).
- *Unused public building:* Previous usage.

5.3.7 Background Information

In addition to the classes that represent the elements surveyed, the application schema also includes classes that represents geographic information that is used as background elements in the map. These classes are shown in Figure 5.9.

An element that provides important background information are the contour lines, which are used to display the height of the elements in the maps. These lines are further classified in *coast line*, *main contour lines*, *intermediate contour lines*, and *depression contour lines*. Another element that provides important context information is the hydrography. It consists of the rivers and lakes represented as lines and surfaces and identified by its name. Other geographic elements that provide important context information are *railroads*, *walls* and *fences*, *buildings*, *text labels* and *landmarks*. Finally, roads that are not surveyed are also represented as background information.

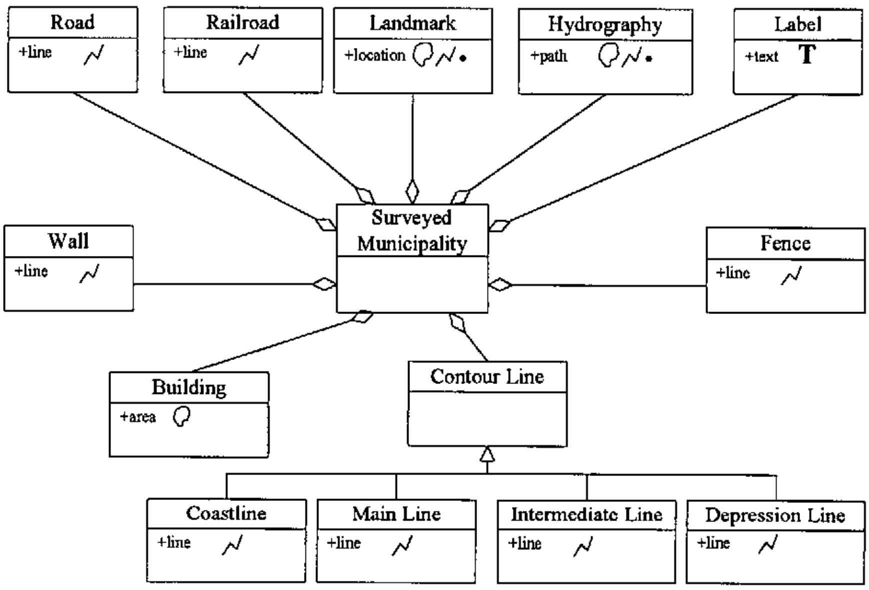


Figure 5.9: Background.

5.4 The EIEL Project Workflow

Considering the vast amount of information that had to be collected to build the EIEL GIS, it was necessary to design a workflow that enables the supervisors to control the process of collecting the information, introducing the data into the database, and validating the correctness of the database. Figure 5.10 shows a diagram that illustrates the workflow. Three main tasks were identified (depicted as thick rectangles): the *database creation* task, the *data exploitation* task, and the *data maintenance* task. Within the database creation task, five subtasks were identified (depicted as thin rectangles). In this section, we describe these tasks and the applications that were built to support each task.

The database creation task involved the creation of the database that stores the data collected by the survey process. The task was divided into five subtasks that were carried out concurrently, and we developed an application to support each of these tasks, namely: a *geographic data import* application, *a geographic data input* application, an *alphanumeric data input* application, a *geographic-alphanumeric linkage* application, and a *data validation* application. We describe these tasks and applications in more detail in Section 5.4.1.

The data exploitation task involves the exploitation of the information collected by the survey. This is mainly done by generating thematic maps and statistical reports.

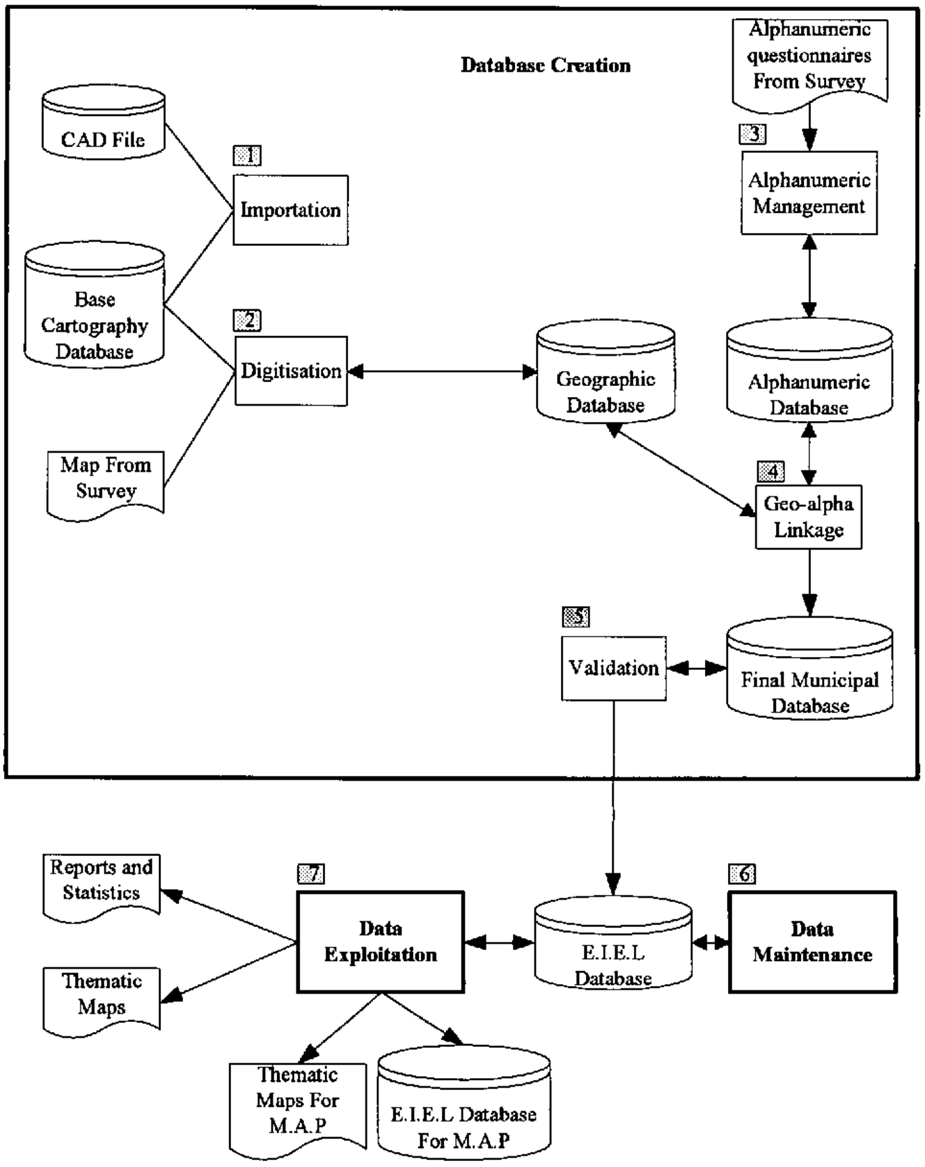


Figure 5.10: Development Process Workflow.

This task is carried out by two different applications, a desktop application deployed that is used by the Provincial Council staff (called *SIGEIEL*, and a web-based GIS that can be accessed over the Internet (called *WebEIEL*). Both applications are briefly described in Section 5.4.2. The *WebEIEL* is described in more detail in Section 5.5.

Finally, the data maintenance task involved the updates to the database performed by the Provincial Council once the survey is completed. These updates may be caused by errors detected in the survey, or by changes in the surveyed items that require changes in the database to keep it up to date. This task is performed used the *SIGEIEL* application, which is described in 5.4.3.

5.4.1 Application for the Creation of the Database

The database creation task involved the survey of the municipalities in the province, the input of the data in the database, and the validation of the correctness of the information. It was decided that the information survey and the data input processes were going to be carried out simultaneously. That is, while a group of people was surveying one municipality, another group was introducing into the database the information from another municipality which had been surveyed earlier. This would cause inconsistencies in the database during the data input process because municipalities are not independent with respect to the elements surveyed (e.g., roads or water supply networks involve more than one municipality). However, it was decided that these inconsistencies could be detected at a later step in the workflow, where they would be resolved.

Therefore, the process of building the database consisted of three steps. First, the information about the element in the survey had to be collected. This step was carried out by students that visited the municipalities and used paper maps and forms to collect the data by interviewing the responsible staff in the municipality or by direct observation. Then, the second step was to introduce the information of the municipality into the database. The third and final step was to validate the information in the database once all municipalities were surveyed to detect inconsistencies or errors caused by collecting and introducing the information of the municipalities separately.

Given that appropriately surveying an element (e.g., a water treatment plant, or an urban plan) requires particular expertise in that engineering field, it was decided to split the information to be surveyed into three disjoint categories that were surveyed by students with different educational backgrounds supervised by professors or experienced students from that field. The categories were the following:

- *Water cycle information.* It consisted of the information described in the application schemas shown in Figure 5.5, Figure 5.6, and Figure 5.7.
- *Infrastructure information.* This category consisted of the information described in the application schemas shown in Figure 5.3, Figure 5.4 and the information in

Figure 5.2 that was not concerned to the territorial structure of the municipality (e.g., power plants, gas network).

- *Territory and equipment information.* This category comprised the application schema in Figure 5.8, and the territorial information in Figure 5.2.

Considering that working with geographic information requires a different training from the training needed to work with the information in the previous three categories, it was decided that the manipulation of geographic information was to be performed by a fourth group of students that were also responsible for collecting the information represented by the application schema displayed in Figure 5.9. The Provincial Council of A Coruña already owned CAD-based cartography of the municipalities. Therefore, it was not necessary to survey the geographic information completely. Rather, it was necessary to perform a process of data import from the CAD-based cartography, and a process of correction and improvement of the imported information.

The separation of the survey responsibilities into four categories caused that the information was entered into the database from four different flows. Geographic information was introduced by the *cartography group*, water cycle information was introduced by the *water group*, infrastructure information was entered by the *road group*, and territorial and equipment information was introduced by the *equipment group*. Particularly, geographic and alphanumeric information of a surveyed element came from two different data flows. Therefore, it was necessary to perform a process of data linkage that built a single data entity by joining the information from both data flows.

As a summary, the task of creating the database was divided into five tasks that were carried out concurrently. On the one hand, CAD-based cartography had to be imported and adapted to the GIS database. This was performed by the *geographic data import* task. The geographic information in this database was used to create paper maps that were used by the surveying teams in their field work. When these teams returned with the information surveyed, the geographic information and the alphanumeric information collected was introduced into the database. These tasks were performed within the *geographic data input* task and the *alphanumeric data input*. Once these two tasks were finished, a single entity was created for each surveyed element in the *geographic-alphanumeric linkage* task. Finally, the consistency and correctness of the information in the database was checked in the *data validation* task.

Our work during this process included designing and creating an application with different modules that supported these tasks. In the following sections, we briefly describe these modules.

Geographic Data Import Module

The first step in the workflow was to create a *cartography database* that could be used as the base for the survey and the data input process. For this purpose, the digital cartography owned by the Provincial Council was used. However, the digital cartography was created and stored as CAD files. Therefore, it was necessary to create a module in the application to transform the information on these files and to adapt it to the requirements of a geographic database.

Particularly, geographic objects were not appropriately represented in the CAD file. The most important problem that we found in the CAD-based cartography was that data representation and data visualization were not separated. Instead, the information in the cartography was represented in the way that it was visualized.

As an example, the common visualization style for non-asphalted road sections is to draw one of the sides of the road as a broken line. If representation were separated from visualization, the road would be drawn as a double continuous line, and one of the lines would be drawn broken only for visualization purposes. However, the road was represented in the CAD-file as a continuous line and a collection of line segments for the broken line.

Another common problem caused by not separating the representation of information from its visualization is that graphical objects were not completely drawn because they were hidden by other graphical objects. For instance, if the border of a road was coincident with the border of building or of a wall, one side of the road was not drawn because it was not going to be displayed, as it was hidden by the building or the wall. Therefore, the objects included in the CAD-file cannot be used directly in the application. New geographic values had to be drawn using the existing objects in the CAD-file as guidelines. The geographic data import module was designed to facilitate this task.

Another problem that we found in the CAD-based cartography was that single objects of the application schema were drawn using multiple graphical objects. For instance, contour lines were not represented with a single line, but as a collection of independent segments. Moreover, the height of the contour line was not a property of the object. Instead, it was a graphical label attached to the object. Hence, a tool was necessary to enable the user to build a contour line by selecting a collection of connected but independent strokes, and associating a height value to the object by entering it in a dialog box.

A final problem that had to be solved, was that the graphical object used to represent a geographic object was usually of a wrong data type. For instance, municipality borders were usually represented as curves in the CAD-files, whereas they were represented as surfaces in the geographic database. Therefore, the geographic data import tool module had to enable a user to create surfaces from collections of lines, possibly by correcting the points where the surfaces were not correctly closed.

GIS development tools often implement functionality to perform these tasks, but it is provided in a generic way that is not easy to use by an end-user. Our work included customizing the GIS development tool to import the CAD files existing at the Provincial Council, and providing easy-to-use commands to perform the corrections on the geographic data.

Geographic Data Input Module

Once the groups in charge of collecting the information returned from the municipalities, the people of the cartography group used the annotations on the paper maps to draw the geographic objects surveyed by the groups in the municipality. The new database resulting from this step was composed by newly-drawn objects (e.g., road sections, hospitals, or water pipes), and objects that were copied directly from the cartography database (e.g., contour lines, buildings, hydrography elements, or railways).

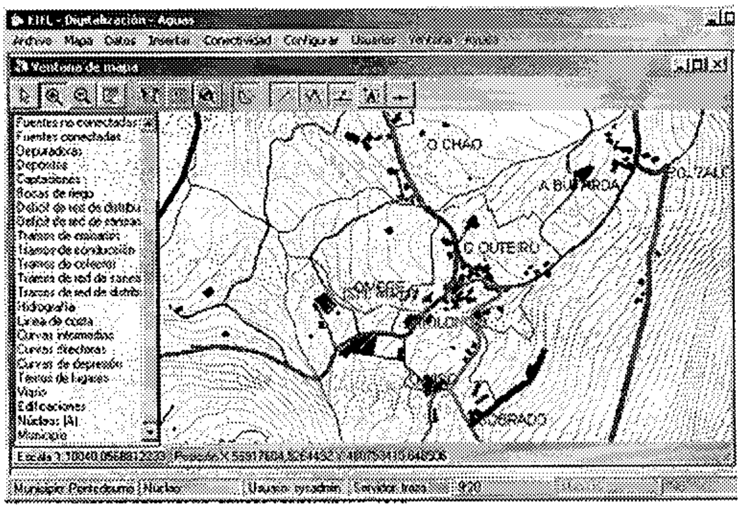

EIEL - Digitalización - Augas
Arquivo Mapa Datos Insertar Conectividad Configurar Usuarios Ventana Axuda
Ventana de mapa
O CHAO
A BURROA
O OUTEIRO
OBRADO

Figure 5.11: Geographic Data Input Module.

For this task, we developed a module for the application that allows users to draw geographic objects from scratch and to copy geographical objects from the cartography database to the geographic database. Figure 5.11 shows a screenshot of this module.

The advantages of developing a custom-made module for this task instead of using directly the GIS development tool user interface can be summarized as follows:

- *A user interface adapted to the application schema.* Instead of using the generic user interface of the GIS development tool, the user manipulates a user interface that is tailored to the application schema. For instance, the menu option to load additional information in the GIS development tool is called *Add Layer*, and presents a list of all available feature collections. In our application, the user is directly presented a menu that uses the names of the application schema classes grouped into meaningful categories (e.g., a top-level *infrastructure* menu that contains the entries *road section, street section* and other similar application schema classes).

- *Data consistency checking.* In addition to the generic consistency checks implemented by the GIS development tool (e.g., surfaces must be closed), it is possible to implement consistency checks derived from the application schema (e.g., a street section associated to a population center must be contained within the boundary of the population center).

Alphanumeric Data Input Module

Similarly to the geographic data input module, it was necessary to design and implement modules to allow users to input in the database the alphanumeric information that was collected by the surveying groups. Particularly, we developed one module for each group. Figure 5.12 shows a screenshot of one of these modules.

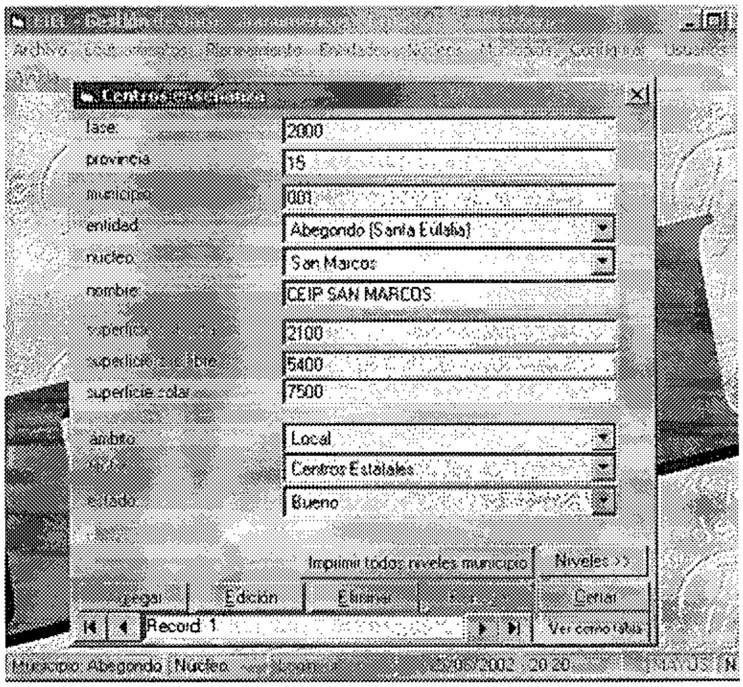
The data entered using these applications is stored in an alphanumeric database. The advantages of developing these modules instead of using the GIS development tool directly are the same as in the previous case: the user interface is tailored to the application schema, and data consistency checks can be implemented in addition to those enforced by the GIS development tool.

Geographic-Alphanumeric Linkage Module

The workflow forced the geographic data of a surveyed element to be entered separately from the descriptive attributes. Once all data of a municipality were entered, it was necessary to join these two parts of each surveyed element to build the final geographic object.

We designed and implemented a module to support this task, which can be seen in Figure 5.13. The user interface displays on one side of the screen a map with the geographic objects and a table with the alphanumeric data. In order to create the surveyed data element, the user only has to select for each table row the corresponding geographic data item. In the other side of the screen, the user interface displays the geographic objects that have already been built in order to allow to undo changes.

Sometimes, this process could be done automatically. For instance, if there was only one hospital in the municipality, it was clear how to create the surveyed data

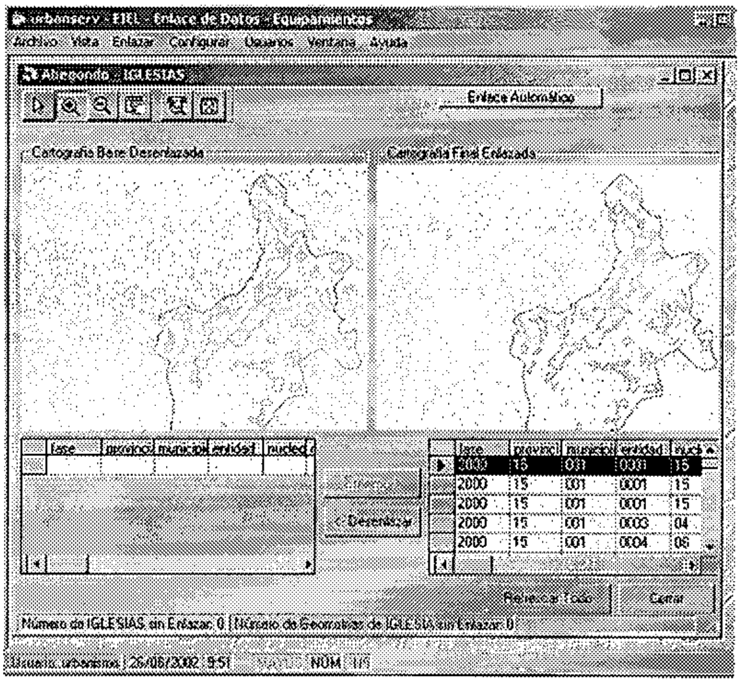


fase	2000
provincia	15
municipio	001
entidad	Abegondo (Santa Eulalia)
nucleo	San Marcos
nombre	CEIP SAN MARCOS
superficie [illegible]	2100
superficie [illegible] libre	5400
superficie solar	7500
ambito	Local
[illegible]	Centros Estatales
estado	Bueno

Record 1

Municipio: Abegondo Nucleo: [illegible] 25/06/2002 20:20

Figure 5.12: Alphanumeric Data Input Module.


urbanserv - EIEL - Enlace de Datos - Equipamientos
Archivo Vista Enlazar Configurar Usuarios Ventana Ayuda
Abegondo - IGLESIAS
Enlace Automático
Cartografía Base Desenlazada
Cartografía Final Enlazada
Enlazar
<- Desenlazar
Refrescar Todo
Cerrar

fase	provinci	municipi	entidad	nucl
2000	15	001	0001	15
2000	15	001	0001	15
2000	15	001	0001	15
2000	15	001	0003	04
2000	15	001	0004	06

Número de IGLESIAS sin Enlazar: 0 | Número de Geometrías de IGLESIAS sin Enlazar: 0

Usuario: urbanismo | 26/06/2002 9:51 | NUM | 1/1

Figure 5.13: Information Linkage Module.

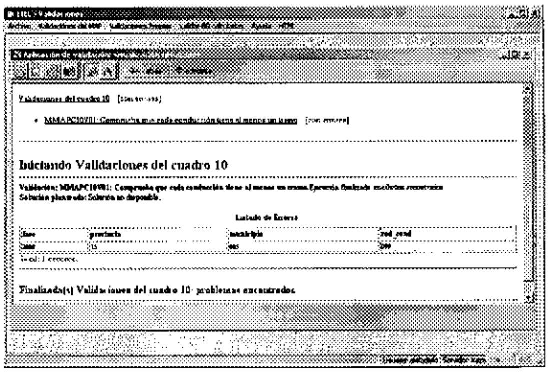
element. In order to facilitate the task, the module automatically detected these cases and created the surveyed data elements under the user supervision.

Data Validation Module

The result of the linkage process is a database that is structurally equal to the final database of the information system. However, the consistency of the information could not be checked completely because it was not possible to introduce the data for all municipalities at the same time. Given that the data collection and the data input tasks were carried out simultaneously, the data of neighboring municipalities were entered with a time difference of months between them.

As a result, a final validation task had to be carried out before the data was moved to the final database. This task implied checking the data against a set of validation rules developed by the Spanish government, which were extended by the data collection groups, to ensure the quality of the data. Moreover, our group developed an additional set of rules to ensure the correctness of the geographic information (e.g., the geographic object that represents the border of a population center must be within the borders of the municipality).

We developed a module that performed this validation task and created reports with the results. A screenshot of this module can be seen in Figure 5.14. Once all validation rules were fulfilled by the data, the application enabled the user to move the information to the final database.



Validaciones del cuadro 10 [illegible]

- MMAPC10V01: Comprueba que cada conducción tiene al menos un tramo [illegible]

Iniciando Validaciones del cuadro 10

Validación: MMAPC10V01: Comprueba que cada conducción tiene al menos un tramo [illegible]
Solución planteada: Solución no disponible.

Listado de Errores

fase	provincia	municipio	red_cond
[illegible]	15	[illegible]	[illegible]

Total: 1 errores.

Finalizada(s) Validaciones del cuadro 10: problemas encontrados

Figure 5.14: Validation Module.

5.4.2 Applications for the Exploitation of the Database

Once the geographic database was complete, and the information contained within was validated, it was necessary to design and implement applications that enable to use the information stored in this database. These applications had to provide at least the following functionality:

- *Creation of thematic maps.* The application must enable defining maps from the information of the database. These maps may be visualized and inspected interactively on the screen, printed as paper maps, or exported to image or CAD files. In addition to these maps, a set of predefined maps must be available, which can easily be used without having to be defined.
- *Creation of reports.* The information to be included in these reports ranges from a simple enumeration of the elements in the municipality to complex statistical data analysis. Similarly to the creation of maps, the user may define his own reports or use predefined reports.
- *Generation of the database for the Spanish government.* We cannot forget that the main goal of this project was the fulfillment of the survey required by the Spanish government. The application had to allow the staff at the Provincial Council to produce the database and the set of thematic maps required by the Spanish government.
- *Publishing the data over the Internet.* In order to enable the politicians and technical staff at the municipalities to access the information collected by the survey, it is important to offer a web-based application to query and visualize the information in the database. Moreover, the inhabitants of the province (in fact, anybody connected to the Internet) can use this application to visualize information about the province.

In order to deploy a single application at the Provincial Council, we decided to include the functionality described in the first three items of the previous enumeration in the database maintenance application. We describe this application (called *SIGEIEL*) in Section 5.4.3. On the other hand, given that we have tested the architecture proposed in Chapter 4 in the web-based application (called *WebEIEL*), it is described in more detail in Section 5.5.

5.4.3 *SIGEIEL*: Maintenance of the Database

The database maintenance task involves the data insertions, updates and deletions performed by the staff at the Provincial Council. These changes may be caused by two reasons:

- *Changes in real-world objects.* When a surveyed entity of the real-world changes, the staff at the Provincial Council must change the database to keep it up to date. For instance, if a water supply pipe section is substituted with a new one, the information in the database must be updated.
- *Errors in the survey process.* If the staff at the Provincial Council detects errors in the information, they must be able to modify the information in order to correct these errors.

We have designed and implemented an application to be deployed at the Provincial Council to perform these tasks. The functionality implemented in some of the modules in the application developed for the database creation task is used in this application:

- *Geographic data input module.* The functionality implemented in this module is used to manipulate the geographic attributes of the database.
- *Alphanumeric data input modules.* The same forms that were used to input the alphanumeric data in these modules is used in the maintenance application to manipulate the descriptive attributes of the database.
- *Data validation module.* After the data in the database is modified, it is necessary to check the consistency of the information using the validation rules implemented for the data validation module.

In addition to this functionality, the data exploitation functionality described in Section 5.4.2 is also included in this application. This implies that it is necessary to implement a user authentication procedure in the application because it can be used only to visualize information, but also to modify it. Hence, it is important to ensure that only authorized users modify the information in the database.

5.5 *WebEIEL*: Web-based Exploitation Application

This section describes in more detail the *WebEIEL* application, which was developed following the architecture proposed in Chapter 4 using commercial GIS development tools. First, in order to give an idea of the functionality of the application, we summarize its user manual in Section 5.5.1. Then, we describe the architecture of the application in Section 5.5.2.

5.5.1 The *WebEIEL* User Interface

The contents of the *WebEIEL* site are organized into five main sections:

- *Home page.* It is the entry point to contents of the web application. It allows the user to select the application language, and shows the main page of the site after having agreed with the conditions of use (see Figure 5.15).
- *Thematic maps.* These maps show all the information about the province using interactive thematic maps (e.g., road network, sewerage, or water supply). Users may select a given area to be displayed, and browse the information of any of the infrastructures or urban services displayed.
- *Synthetic maps.* These maps display a choropleth map of the state of the service in the municipalities of province, using the different synthetic indicators defined by the Spanish Ministry of Public Administrations.
- *Downloadable maps.* This section allows the user to download for each municipality, and for each pre-defined thematic map, a PDF file containing containing a map series of the municipality and its population centers at a fixed scale.
- *Reports.* From this section, the user can download PDF files that contain textual information about the infrastructures and urban services of the selected municipality
- *Comments.* It allows the user to send comments to the staff at the Provincial Council.
- *Work team.* It displays the people that worked at this project.

In the following sections, we describe separately each of the nodules of the application.

Thematic Maps

In order to display an interactive thematic map for an area within the province of A Coruña, the user must select in the application:

- *The type of map.* Six thematic maps are pre-defined (i.e., urban planning and service provision, road network, road pavement, water supply, sewerage, and urban services). Additionally, a customized thematic map can also be built from scratch.
- *The display area.* The application lets the user choose the initial display area of the map: the complete province, a municipality, a population entity, or a population center.

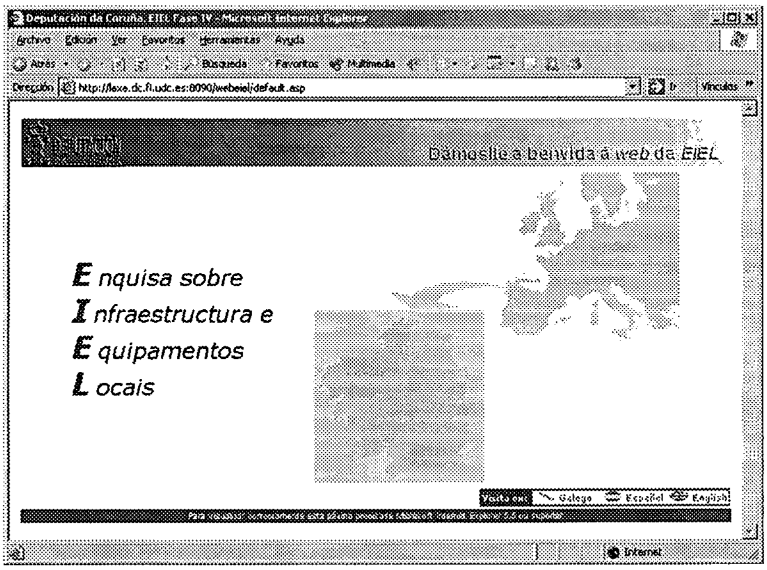

Deputación da Coruña. EIEL Fase IV - Microsoft Internet Explorer
http://lexe.dc.fi.udc.es:8090/webeiel/default.asp
Dámoslle a benvida á web da EIEL
E nquisa sobre
I nfraestructura e
E quipamentos
L ocais
Galego
Español
English

Figure 5.15: *WebEIEL*: HomePage.

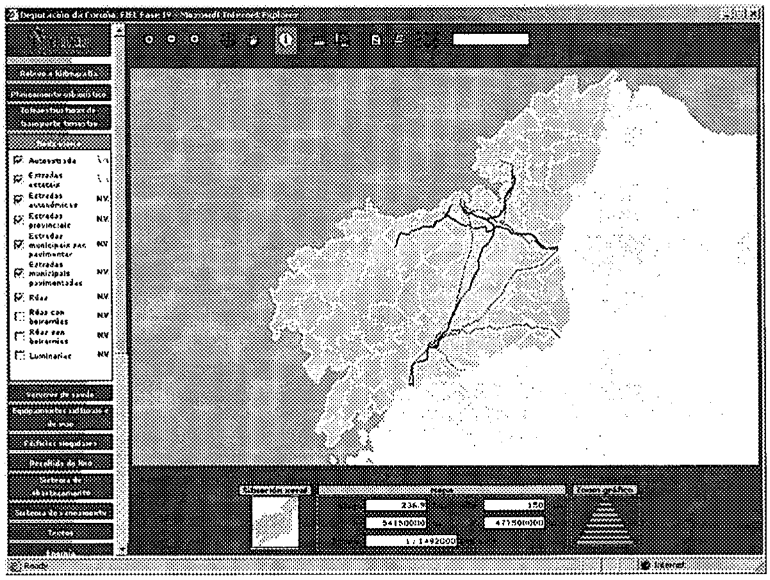


Figure 5.16: *WebEIEL*: Map Window.

- *Map size.* In order to support different connection speeds, the size of the map can also be selected. The smaller the map resolution, the smaller the size of the file to be transmitted.

After performing these selections, the user is presented a map window similar to the one depicted in Figure 5.16. The window is divided into four areas:

- *Graphical legend.* It is located on the left side of the window, and it enables the user to select the information to be displayed on the map. Cartographic layers are grouped into categories to facilitate the visualization. The user can configure whether the cartographic layer must be visible or not using a check box located close to the layer name. The application performs automatic map generalization, and controls the type and resolution of the geographic object displayed for the cartographic layer at a given map scale. The application even controls whether a cartographic layer should be visible at all at a specific map scale.

- *Tool bar.* It is located on the upper side of the window. The user can interact with the map using the tools in this bar. Particular, the user can *zoom*, *center* and *pan* the map view, *get information* of a geographic feature by clicking on a point in the map, *measure* lines and rectangles in the map, *reload* the map after changing the layers that are visible, and *print* the map to a PDF file.

- *Context information.* It is shown at the lower edge of the page, and it displays an *overview map* that shows the are of the province that is being visualized, the map width, height, center coordinates, and scale, and a graphical representation of the zoom level that can be used to change the map scale in a visual way.

- *Map area.* It is the central area of the window, and displays the current map. It also displays a tool tip with information of the geographic features when the mouse is moved over the geographic objects.

The map tools and the graphical legend allow the user to build a customized view of the information, to print this view, and to retrieve information of the displayed objects. Figure 5.17 shows an example of a map window customized.

Synthetic Maps

To visualize a choropleth map displaying the values of one of the indicators, the user has to select in the application:

- *The indicator type.* The user selects the indicator to be displayed from a list.

- *The map size.* This selection enables the user to choose the map size more appropriate for its display size.

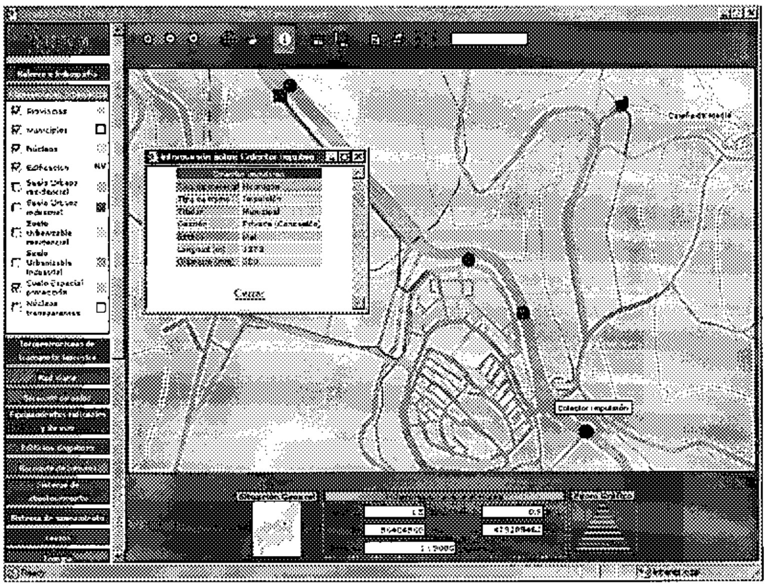


Figure 5.17: *WebEIEL*: Map Navigation.

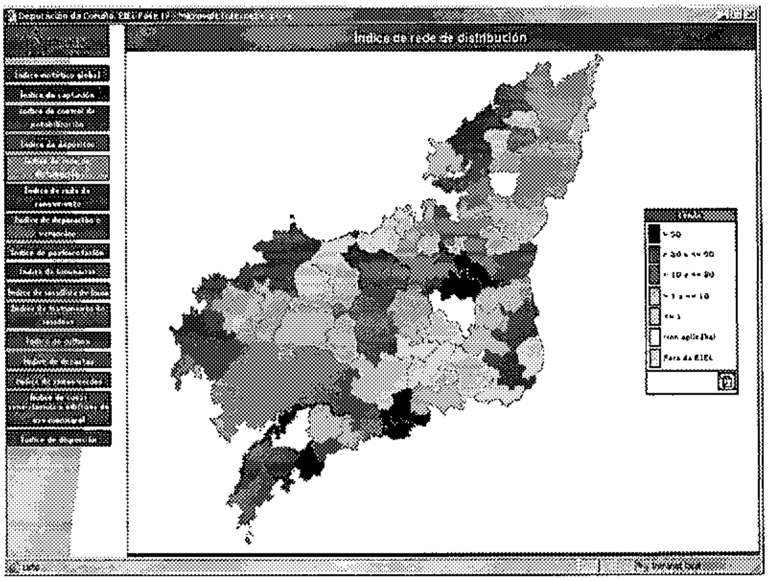


Figure 5.18: *WebEIEL*: Synthetic Map Window.

After giving values to these parameters, the application shows a synthetic map window similar to the one shown in Figure5.18. The window shows a map of the province with each municipality colored according to the value of the indicator. The actual value of the indicator for each municipality is displayed when the user moves the mouse over the map. The graphical legend of the map is displayed on the right of the map. The user can print the map to a PDF file by clicking on the button at the bottom of the graphical legend. Finally, the user can select a different synthetic map using the buttons on the left side of the window.

Downloadable Maps and Reports

The thematic map and the synthetic map allow the user to print the maps that are being displayed. However, it is also necessary to provide a means to print a completely municipality at a fixed scale. Similarly, feature information can be retrieved in the thematic map window, but a way is needed to get all the information of a given feature type for a municipality. The downloadable maps and reports sections allow the user to download PDF files with this information. For each of the pre-defined thematic maps, and for each municipality, two files are available: one with the complete municipality using the scale 1:10,000, and another with the population centers in the municipality using the scale 1:5,000. Similarly, for each municipality, each of the reports pre-defined by the Spanish Ministry of Public Administrations can be downloaded.

5.5.2 System Architecture

We tried to apply the architecture proposed in Chapter 4 for the implementation of the *WebEIEL* application. This enabled us to prove the applicability of our proposal in a real-world problem. However, there were time and cost requirements for the implementation that forced us to use existing commercial applications instead of developing new software components from scratch.

Figure 5.19 shows the system architecture of *WebEIEL*. The components that we had to develop are shown with a gray background, whereas the commercial components are shown with a white background. The dashed lines represent the separation of the tiers of the architecture. The system is web-based, and therefore the functionality is structured using a client-server architecture.

On the server side, the information is managed by a relational DBMS (Microsoft SQL Server 7.0). The geographic information is managed by Intergraph Geomedia Web Map, which follows the opaque relational approach using BLOB attributes in the relations to store the geographic values. The activity module, and the style and generalization module had to be developed particularly for this application because the ones provided by Geomedia were not sufficient to meet our requirements. Finally, the information of the web application is served by Microsoft Internet Information Server,

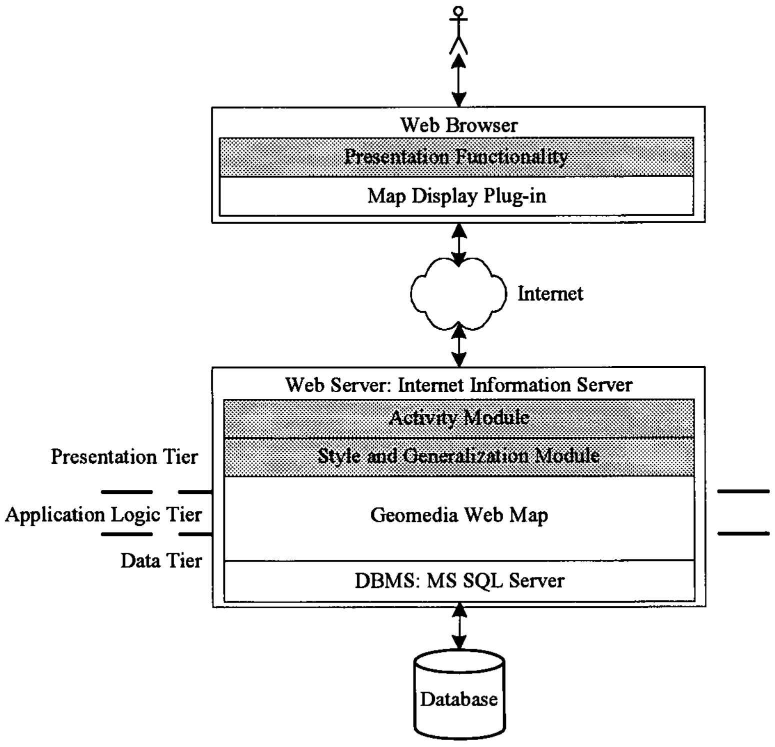


Figure 5.19: System Architecture for the EIEL GIS.

the web server supplied by Microsoft, which was chosen because it is required by Geomedia Web Map.

The client side of *WebEIEL* was implemented using dynamic HTML, that is, web pages with scripts implemented in Javascript to provide client-side functionality. The map image is displayed by a map display plug-in embedded in the web page. This plug-in was developed by Intergraph, and it is a Java applet that displays the map image, and implements many presentation functionality such as the computation of distances and areas. Therefore, the *WebEIEL* application uses a *thick-client* approach (see Section 2.6.2).

The main reasons for choosing a thick-client approach for the client were these:

- A thick-client approach makes easier to provide the user with some interactive functionality. In our particular case, we wanted the application to display a tool tip message when the mouse is over a geographic object to identify the geographic feature. This is not an easy task to do using only dynamic HTML, but it is much easier when a thick-client is used.

- To display the map using a vector image format that provides a much higher quality than raster image formats. There is no native support for vector image formats in web browsers, and therefore, a thick-client is needed.

5.6 Summary

In this section, we have presented the EIEL project [BCF$^+$01a, BCF$^+$03], which is a complex GIS application developed for the Provincial Council of A Coruña by a group of five people during three years. We have described the application schema of the GIS application, the workflow that we followed to carry out the project, and the applications that were developed to support this workflow.

The experienced gained during the development of this project, as well as other previous GIS projects [BCF$^+$01b, BCLV02, ABF$^+$02, ABF$^+$03], has served as inspiration for the architecture proposed in Chapter 4. However, since we have used commercial GIS development tools for the development of the applications described in this chapter, they have the problems caused by the use of these tools. In the next chapter, we analyze these problems is detail.

Chapter 6

Problems of GIS Applications Created Using GIS Development Tools

6.1 Introduction

We have described in Chapter 5 the EIEL project, a complex geographic information system that we have developed for the Provincial Council of A Coruña as part of the work for the PhD Thesis. Two GIS applications are included as components of this geographic information system: *SIGEIEL*, which is an application for the maintenance and analysis of the information in the database, and *WebEIEL*, which is a web-based application for the visualization of the information contained within the database.

We have tried to apply in the development process of these GIS applications the architecture and the guidelines that we propose in Chapter 4. However, special time and cost requirements for the project forced us to use existing GIS development tools, instead of implementing the applications from scratch using custom-developed modules. The result of using GIS development tools to create *SIGEIEL* and *WebEIEL* is that the architecture of the system differs in many aspects from the architecture we propose in Chapter 4, loosing some of the properties of our architecture (e.g., flexibility, or reusability).

In this chapter, we compare the architecture of *SIGEIEL* and *WebEIEL* with the architecture that we propose in Chapter 4, and we analyze the causes and consequences of the differences. We also show that these differences are common to many GIS applications developed using GIS development tools. The rest of this chapter is structured as follows. First, in Section 6.2, we compare the architecture of the applications developed for the EIEL project to the architecture we proposed in Chapter 4. Then, in Section 6.3 we describe the most common problems that can be identified in GIS applications implemented using GIS development tools. We follow with a detailed analysis of the differences in each tier of the architecture. Section 6.4 analyzes the data tier, Section 6.5 analyzes the application logic tier, and Section 6.6 analyzes the presentation tier. Finally, we summarize the chapter in Section 6.7.

6.2 Comparison of the Architectures

In this section, we present a comparison between our architecture proposal and the architecture of the *WebEIEL* application, which was developed using the GIS development tool Intergraph Geomedia Web Map.

Figure 6.1 shows again the architecture that we have proposed in Chapter 4. Our proposal separates the functionality of GIS applications into three independent tiers, namely the *Data Tier*, which provides data management independently from the software technology, the *Application Logic Tier*, which implements the problem-solving and the application-specific functionality of the system, and the *Presentation Tier*, which is responsible for data visualization. Each tier is further decomposed into software layers, which are described in Chapter 4

Figure 6.2 displays a detailed view of the architecture of the *WebEIEL* application. The components that we had to develop are shown with a gray background, whereas the components provided by Intergraph Geomedia Web Map are shown with a white background. The figure also shows the software layers of our proposal that are equivalent to the components of the *WebEIEL* application.

It can be seen that both architectures follow the traditional three-tier architecture design. However, the main difference is that the tiers of our architecture proposal are completely independent, they only interact at the interfaces, whereas the tiers of an application developed using Intergraph Geomedia Web Map are implemented in a single monolithic software module. This causes flexibility and reusability problems, which are described in Section 6.3. We represent the monolithic software module in the figure using the big white rectangle called *Geomedia Web Map* that contains all the functionality provided by Intergraph for the different tiers of the architecture.

Another difference between our architecture proposal and the architecture of GIS applications developed using Intergraph Geomedia Web Map is that some of the functionality within the tier of our architecture is not implemented, and it must be provided by the developer (e.g., the activity module, or the style module).

Now, we compare each of the tiers in the architecture in more detail:

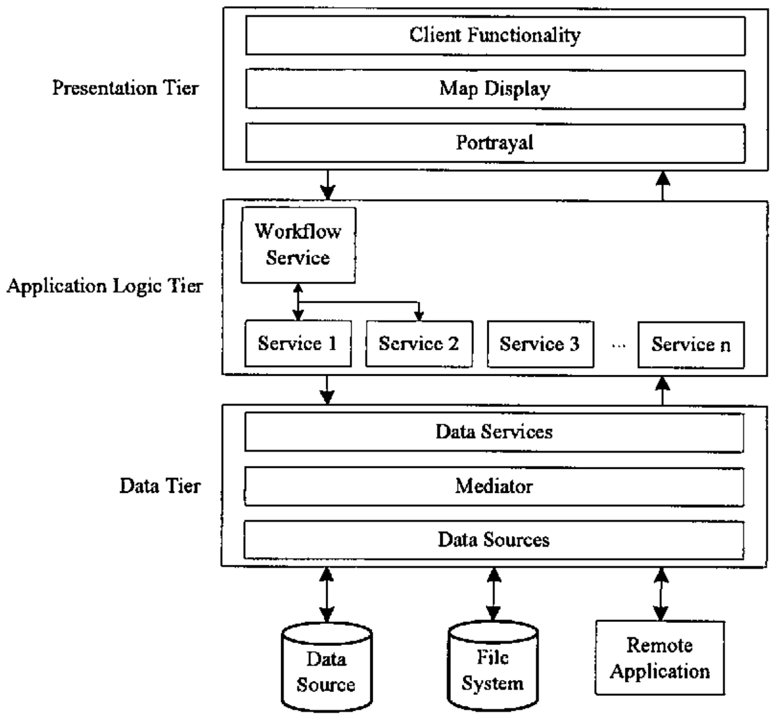


Figure 6.1: Software Layers of our Architecture Proposal.

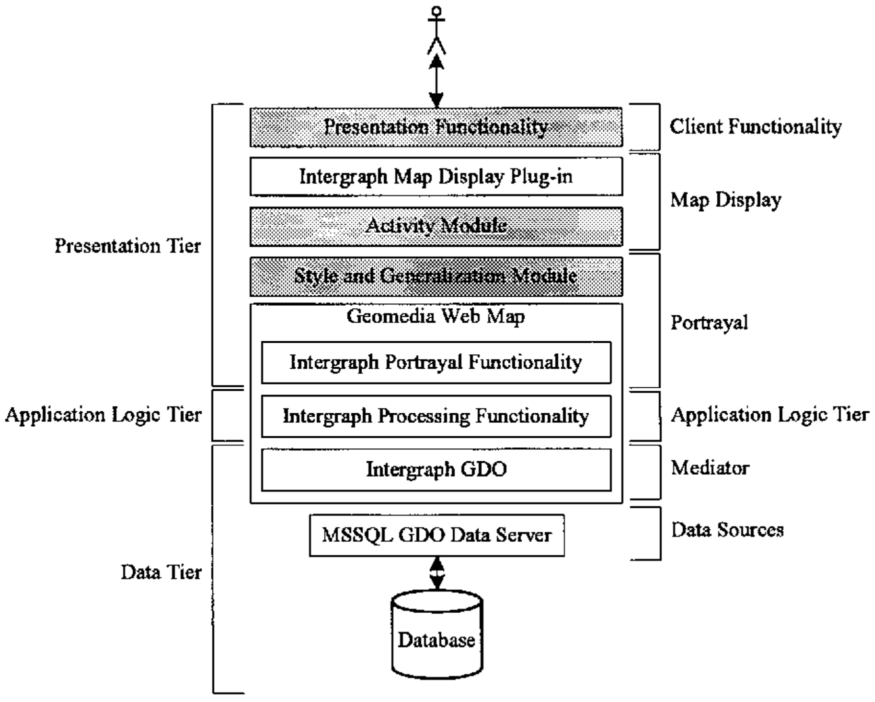


Figure 6.2: Software Layers of *WebEIEL*.

- *Data tier.* GIS applications developed using Intergraph Geomedia Web Map use a mediator-wrapper pattern for geographic data management using the proprietary technology called *Intergraph Geographic Data Objects (GDO)*. The GDO layer provides a single conceptual model for geographic information, which is stored in its native data source. A separate *GDO server* is provided by Intergraph for each data source type. This is similar to our proposal of a mediator layer and a data sources layer (see Section 4.4.1, and Section 4.4.2).

  However, the architecture of GIS applications implemented using Intergraph Geomedia Web Map does not include a data services layer, as proposed for our architecture (see Section 4.4.3). These services must be implemented by developers with no support from Intergraph Geomedia Web Map.

- *Application Logic Tier.* In our architecture proposal, this tier is composed of a collection of independent services that are responsible for a simple and well-defined task (see Section 4.5. On the other hand, in applications developed using Intergraph Geomedia Web Map, this tier is implemented within a single software module that provides all the functionality. As we said before, this monolithic approach causes many flexibility and reusability problems.

- *Presentation Tier.* We proposed in Section 4.6 the division of the functionality of this tier into three independent layers. The architecture of GIS applications implemented using Intergraph Geomedia Web Map also divides the presentation tier into these three layers, but much of the functionality that we proposed in Section 4.6 is missing. Therefore, the developer must provide custom implementations of this functionality, as we did for the *WebEIEL* application.

Summarizing, the main differences between our architecture proposal and the architecture of GIS applications developed using Intergraph Geomedia Web Map are:

- The functionality of the tiers is implemented in a monolithic software module in applications developed using Intergraph Geomedia Web Map, which reduces flexibility and reusability.

- Some of the functionality that we propose in Chapter 4 is not present in GIS applications implemented using Intergraph Geomedia Web Map.

6.3 General Description of the Problems of GIS Development Tools

In this section, we describe the most common problems that can be identified in GIS applications implemented using GIS development tools. First, we analyze these problems in a general manner, and then we show concrete examples that can be identified in the *WebEIEL* application.

6.3.1 Generic Analysis of GIS Development Tools

The architecture of GIS applications developed using GIS development tool does not fulfill many of the general requirements enumerated in Section 4.2. Particularly, the architecture usually fails to meet the following requirements:

- *Flexibility.* The architecture of GIS applications implemented with GIS development tools cannot easily accommodate changes in the requirements. The functionality of the architecture is often implemented in a single module with proprietary interfaces that are highly-interdependent between them. Therefore, it is not possible to change only parts of the system, when the requirements vary, or to use third-party modules for specific problems.

- *Reusability.* Given that GIS development tools often define proprietary interfaces to access their functionality, custom-developed modules for a GIS application implemented using a GIS development tool cannot be used with an application developed using a different GIS development tool. This is because the modules are forced to use the proprietary interfaces, which are not present in other GIS development tools. The result is that the modules are tied to a specific GIS development tool.

These problems are common to many GIS development tools. Traditionally, the main interest of vendors of GIS development tools has been to offer additional functionality as fast as possible, and to capture clients in such a way that they cannot switch to competitors. Building interoperable systems has not been a major concern until a few years ago. Nowadays, the vendors of GIS development tools are realizing that users and developers prefer open and modular tools over closed and monolithic ones.

In addition to not fulfilling some of the requirements described in Section 4.2, applications developed using GIS development tools often suffer from the following problems:

- *Proprietary interfaces.* The data formats and the interfaces to many components of a GIS development tool are often proprietary and private. This causes many problems to developers because they cannot easily implement modules to extend the functionality of applications developed using GIS development tools. For instance, the interfaces to the data sources and the data formats for the representation of geographic information are often proprietary, therefore, the developer is limited to use the data sources that are supported by the GIS development tool.

- *Monolithic software modules.* The functionality of GIS development tools is often implemented using a monolithic software module. Therefore, the architecture of GIS applications implemented using these GIS development tools is also monolithic. This reduces the flexibility and the reusability of the architecture, as described before. For instance, it is not easy to integrate third-party services and modules in monolithic GIS applications because the internal modules of the GIS development tool cannot be replaced. The tiers and the software layers within each tier are highly-interdependent in these GIS applications. As an example, the application logic tier cannot retrieve information from a data source not supported by the GIS development tool, and the presentation tier cannot display information coming from services not provided by the GIS development tool.

Summarizing, the main consequence of these problems is that GIS applications developed using GIS development tools that do not satisfy the requirements imposed to our architecture are too heavily integrated with the technology and cannot be ported to other software platforms. It is extremely difficult for these applications to use functionality provided by other software vendors, and it is very difficult to reuse the functionality implemented specifically for the applications with other GIS development tools.

6.3.2 Examples in *WebEIEL*

The *WebEIEL* application has been implemented using a GIS development tools that suffers from many of the problems described in the previous section. Therefore, it is easy to find concrete examples of these problems in the architecture of *WebEIEL*. For instance, all the additional modules that we have implemented for the application use intensively proprietary interfaces of Intergraph Geomedia Web Map. Therefore, the custom-developed application modules are not reusable.

6.4 Analysis of the Data Tier

In this section, we analyze the problems identified in the data tier of the architecture of GIS applications developed using GIS development tools. Then, we illustrate these problems with examples extracted from the architecture of the *WebEIEL* application.

6.4.1 Common Problems in GIS Development Tools

Figure 6.3 shows a detailed view of the data tier of the architecture proposed in Chapter 4. This tier is composed of three independent software layers: the mediator layer, the data sources layer, and the data services layer. The mediator layer and the data sources layer implement a mediator-wrapper architecture for data management, as described in Section 4.4. The conceptual model for geographic information is defined in the mediator layer, and wrappers for each type of data source are implemented in the data sources layer adapting the specific data source types to the conceptual model. Finally, the data services layer is used to provide multiple different interfaces for the conceptual model defined in the mediator layer.

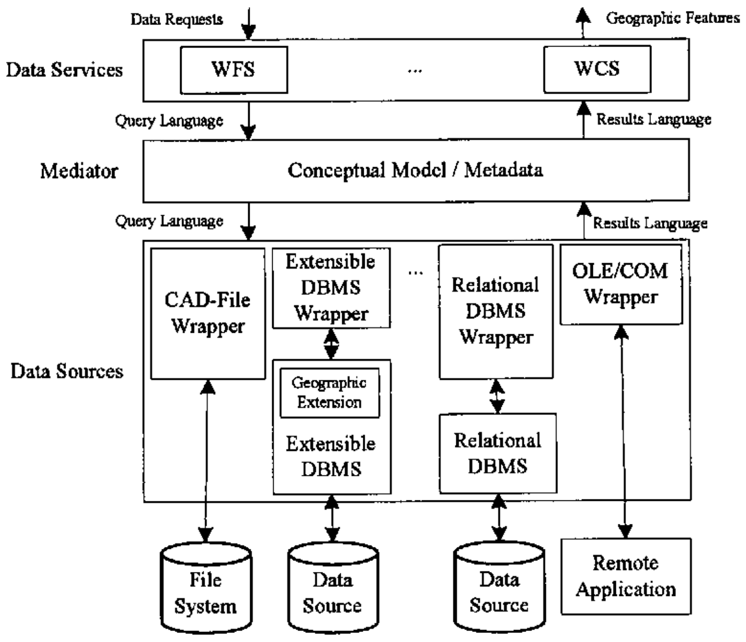


Figure 6.3: Architecture of the Data Tier.

Usually, the data tier of applications implemented using GIS development tools is not designed as proposed in Section 4.4, and therefore, it does not meet the requirements enumerated in Section 4.2. The problems most commonly identified are the following:

- *A proprietary definition of the conceptual model and the logical model.* GIS development tools often define proprietary conceptual and logical models, including proprietary definitions of abstractions for the representation of geographic features, query languages, result languages, and data formats for the representation of geographic values in the data sources. The use of a proprietary query and results language is common on GIS development tools, and a major interoperability problem. Until the publication of standards for spatial query languages [ISO03b, OGC99b], each GIS development tool defined its own proprietary query language. Nowadays, many vendors are working on the implementation of the standard query language for geographic information.

The lack of standards in GIS development tools for conceptual and logical models, query languages, and result languages, causes a problem of technology dependence characterized by the following aspects:

- *Data can only be retrieved through the data tier of the GIS development tool.* The GIS application implemented using this GIS development tool can only use the query language defined by the tool. In order to integrate data from a data source not supported by the data tier with geographic data, the developer must implement an additional module.
- *Data migration is not easy.* Data migration cannot be done directly using DBMS tools because they cannot manipulate the data format for the representation of geographic information. Instead, all the information must be fetched using the data tier of the GIS development tool, and then inserted into the destination data source.

• *No integrated support for the field-based view of space.* It is often the case that the conceptual model defined by GIS development tools does not include integrated support for the field-based view of space. Usually, each conceptual view of geographic space is manipulated using disjoint abstractions that do not allow the developer to perform integrated analysis of information from both views.

• *No topological relationships are explicitly stored.* There is no explicit representation of topological relationships like *these geographic values share this common border*. This causes topological errors in the geographic information of the application developed using the GIS development tool.

• *Operations not provided by the conceptual model.* It may happen that a particular operation required by a GIS application is not provided by the conceptual model of the GIS development tool. If this happens, the developer must implement the operation as an additional module to the application. This often causes an efficiency problem because the operation is implemented outside the scope of the data tier, without using efficient storage and data access structures.

• *No metadata for the information.* Usually, the conceptual model of GIS development tools does not support explicit metadata for geographic information. Therefore, applications implemented using these tools do not allow developers to describe the information stored in the data tier using standard feature catalogues or metadata facilities. Hence, users cannot know the semantics of the information.

• *No data services layer.* We proposed in Section 4.4 that the topmost layer of the data tier must be a layer that contains different data access services for different protocols, or query languages. This layer is not implemented by GIS development tools, and therefore, developers must implement these services with no support from the GIS development tool.

There are GIS development tools that offer solutions to some of these problems. For instance, the object-relational DBMS *Oracle* with the extension module *Spatial Option*, and the extensible relational DBMS *PostgreSQL* with the extension module *POSTGIS* implement the conceptual model defined in Section 3.4.2 [OGC99b] within the scope of the DBMS.

6.4.2 Examples in *WebEIEL*

After having described the general problems that can be identified in the data tier of GIS applications implemented using GIS development tools, in this section we present concrete problems that appeared in *WebEIEL* due to the use of Intergraph Geomedia Web Map:

- The conceptual model does not support integrated querying of field-based information and object-based information. This makes it impossible to analyze simultaneously the road network and satellite-acquired images unless one of the feature types is converted from object-based to field-based or vice versa.

- There is no operation in the conceptual model of Intergraph Geomedia Web Map to compute the center line of a surface. This forced us to explicitly store two different geographic attributes for each road section, its surface and its center line.

- Given that the conceptual model of Intergraph Geomedia Web Map does not support the representation of topological relationships, the common border of municipalities is stored twice, one in each municipality. This has caused that small gaps and overlaps exist between the municipalities because the user cannot digitize the border of the municipality with enough precision.

- The data source wrapper used in *WebEIEL* uses the opaque-relational approach for the management of the geographic values (see Section 2.6.1). That is, geographic values are stored in BLOBs, and the operations are implemented using memory computations. This caused an important efficiency problem.

  The data source wrapper of Intergraph Geomedia Professional for Oracle uses the functionality of Oracle Spatial Option to improve the efficiency of the system. We are currently working on the migration of our applications to this solution. Moreover, we are investigating the use of POSTGIS/PostgreSQL for open-source GIS applications.

6.5 Analysis of the Application Logic Tier

In Section 4.5, we proposed that the functionality in the application logic tier must be implemented by a collection of independent services that are responsible for a simple task and interact through well-defined interfaces. Figure 6.4 shows a detailed view of the architecture of the application logic tier.

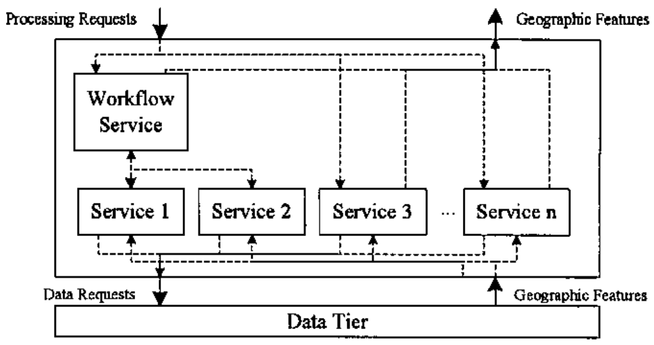


Figure 6.4: Architecture of the Application Logic Tier.

The advantages that this architecture enables are the following:

- New services can be designed, implemented, and added to the architecture whenever needed.
- The architecture is flexible and can be adapted to different runtime environments.

However, the architecture of the GIS applications created using GIS development tools usually implement the whole of the functionality of the tier in a monolithic software module. This causes a flexibility problem, as has been described in Section 6.3.

Moreover, the interfaces defined by the GIS development tools are often proprietary. This causes that custom-developed modules using the application logic tier cannot be reused with another GIS development tools because the interfaces vary. This was described in further detail in Section 6.3.

Finally, the GIS development tool used to create the GIS application many not provide all the functionality required by the application. For instance, the following services are often missing in GIS development tools:

- *Network analysis functionality.* Not all GIS development tools include network-analysis algorithms. If they are not provided, the developer must implement them as a custom-developed module using memory operations, which will probably be inefficient.

- *Simplification service.* In order to perform cartographic generalization at the presentation tier (see Section 2.5.1), it is necessary to simplify the geographic values before they are displayed. However, some GIS development tools do not include functionality to perform this operation. Therefore, the developer must implement this functionality, or use a third-party product.

  Furthermore, if the conceptual model does not support explicit topological relationships, the simplification of geographic values may increase the topological errors in the data. If a common border between two objects is stored in both objects with small differences, the simplification of the border may produce completely different results, hence increasing the errors in the information.

In the particular case of the *WebEIEL* application, we identified the following problems:

- *Intergraph Geomedia Web Map does not include network analysis functionality.*
- *Intergraph Geomedia Web Map does not include a simplification service.* We used an extension module to simplify geographic value. However, the lack of topological relationships in the conceptual model produced many simplification errors. We had to find out the appropriate simplification algorithm and parameters by trial and error.

6.6 Analysis of the Presentation Tier

In Section 4.5 we have proposed a separation of the functionality of the presentation tier into three software layers: a portrayal layer, a map display layer, and a client functionality layer. Figure 6.5 shows a detailed description of the architecture of the tier.

In the following sections, we describe the general problems that can be identified in each tier of the architecture of GIS applications implemented with GIS development tools. Then, we describe the concrete problems that can be identified in *WebEIEL*.

6.6.1 Portrayal Layer

It is often the case that GIS development tools create GIS applications that implement the portrayal layer as a part of the monolithic module that implements also the application logic tier and the data tier. This causes the problems of flexibility and reusability already described in Section 6.3. In addition to this problems, the following functionality is often missing in GIS development tools:

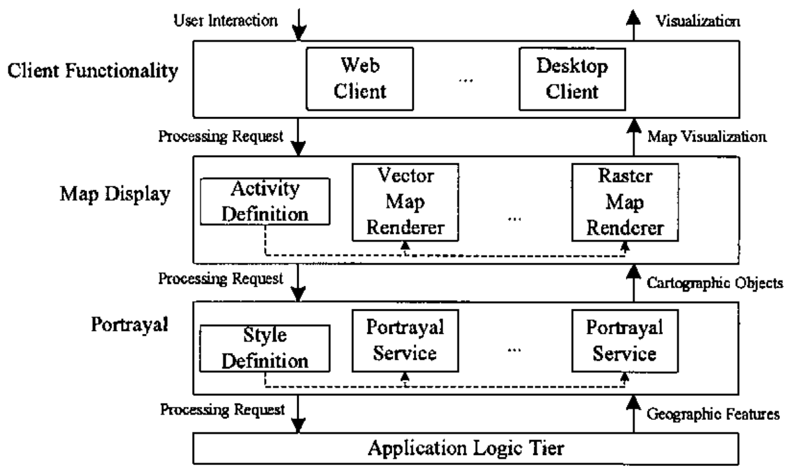


Figure 6.5: Architecture of the Presentation Tier.

- *Complete support for cartographic layers.* We have presented in Section 4.6 an abstraction for the presentation of geographic information that integrates data from different sources. Furthermore, a collection of rules, which use properties of the map and the geographic features, determines the particular geographic features that are displayed and the graphical style that is applied to these features. We have used the name *cartographic layer* for this abstraction.

  This type of cartographic layers is usually not supported by GIS development tools. Instead, a cartographic layer retrieves the geographic values from a single geographic attribute of a single table, and the only property that can be used to decide whether the values are displayed or not is the map scale.

  The consequence is that cartographic generalization (see Section 2.5.1) must be implemented by the developer as an additional module that defines a new abstraction using multiple cartographic layers for the generalization.

  Similarly, a map that classifies geographic features according to the value of an attribute cannot be defined as a single cartographic layer. Instead, so many cartographic layers must be defined as the number of classification ranges.

- *No support for information-density management.* This problem is two-fold. First, it is necessary to control that the amount of information displayed in a map does not overload the physical space available for the map. If the portrayal layer in the architecture of the GIS application does not manage the amount of information displayed in a map, then the result is an overload of geographic objects that cannot be visualized. Secondly, in order to improve the response time of the application, it is necessary to confirm that the size of the portrayed map is not too large. If the GIS development tool does not implement any system to check these two aspects of the resulting maps, they must be implemented by the developer.

- *Limited support for label generation.* The generation of labels for geographic values is a very important functionality that determines the quality of the portrayed maps. GIS development tools often lack the required functionality to create high-quality labels for maps.
- *No support for the production of map series.* An important requirement for GIS applications that is often ignored by GIS development tools is the generation of map series. A map series consists of a collection of maps that partition a given area at a fixed scale. The maps are usually printed using a common page format.
- *No support for the portrayal of reports.* Finally, a report portrayal service must be supported by GIS development tools in order to enable the developer and the user to produce reports of alphanumeric information.

6.6.2 Map Display Layer

In the architecture proposal that we have presented in Chapter 4, the map display layer is composed of map display components that are responsible for rendering the maps produced by the portrayal layer. According to the map output format and the client requirements, a different map display component is selected. For instance, if a web browser is going to be used in the client and if a raster image format is used for the map (i.e., *PNG*, *JPG*), then the map can be rendered directly by the web browser. On the other hand, if the output format selected is a vector format, then a specific map display component must be used because web browsers do not yet support natively any vector image format.

A common problem of GIS development tools is that the support of activity in the map display is not standardized. We have proposed in Section 4.6 that a high-level language must be defined at the map display layer to associate events of the map display component to actions that may invoke the application logic tier. However, as long as this specification does not exist, each GIS development tool defines a proprietary interaction mechanism that results in a lack of reusability in the user interface.

6.6.3 Client Functionality

The final layer of the presentation tier is the client functionality layer that provides the user interface to the end-user. Many GIS development tools create automatically the user interface for GIS applications. However, these user interfaces are often inappropriate for the applications because they are too generic and too close to the user interface of the GIS development tool. The same abstractions and interaction mechanisms that are defined for the user interface of the GIS development tool are directly used in the user interface of the generated GIS application.

6.6.4 Problems Identified in *WebEIEL*

In this section, We describe the concrete problems that we have identified in *WebEIEL* due to use of Intergraph Geomedia Web Map for the creation of the GIS application.

- *There is no support for cartographic layers.* Intergraph Geomedia Web Map does not define a cartographic layer abstraction as we have proposed. This forced us to implement a module supporting cartographic layers.
- *There is no support for cartographic generalization.* Due to the efficiency problem for geographic simplification described in Section 6.5, cartographic generalization cannot be performed dynamically in *WebEIEL*. Instead, multiple versions of the same geographic attribute are computed for each geographic feature at different resolution levels. Then, our module for cartographic layers is used to provide a single abstraction that selects a different geographic attribute according to the map scale.
- *There is no high-quality label generator.* The GIS development tool used for *WebEIEL* (i.e., Intergraph Geomedia Web Map) did not provide such a functionality, and we had to implement a module to generate labels for geographic attributes of type *line* and *surface* that takes into account the orientation of the line or the surface to create a label that is rotated so that it is placed parallel to the geographic object.
- *There is neither a map series portrayal service nor a report portrayal service.* An important requirement for the *WebEIEL* was the generation of map series for each municipality. The application has to divide the surface of the municipality using a fixed-scaled grid, and produce a document with a page for each cell of the grid. The GIS development tool did not provide any support for this kind of portrayal procedure, which had to be implemented as a custom-developed software module. Similarly, Intergraph Geomedia Web Map did not include any service to produce reports for alphanumeric information. We had to implement this functionality ourselves.
- *Problems with the map display component.* Intergraph Geomedia Web Map provides two different map display components for the GIS applications it creates: an *ActiveX component* and a *Java applet*. The main problem of these components is caused by the integration with web browsers. The ActiveX component requires a Microsoft web browser, and causes many problems in the

installation. On the other hand, the Java Applet requires a Java-enabled web browser, and causes less problems in the installation. This is the map display component that we chose for *WebEIEL*.

Another problem is that the data format used for vector images in Intergraph Geomedia Web Map is a proprietary format (called *ActiveCGM*). It is very improbable that this format will be natively supported by web browsers. It is more probable that an open format based on standards like *SVG* will be supported in the near future. Using this language, when it is natively supported by web browsers, would eliminate the problem of integrating the map display component.

6.7 Summary

In this chapter, we have compared the architecture of a GIS application created using a GIS development tool with the architecture that we propose in Chapter 4. We have also presented a collection of general problems that are common to GIS applications developed using GIS development tools, which are caused by the limitations inherent to many of these tools. Furthermore, we have enumerated the consequences of many of these problems in the context of the web-based application that we developed for the EIEL project (see Chapter 5).

The problems caused by GIS development tools can be classified as follows:

- *Technological dependence.* Many GIS development tools use proprietary interfaces for the implementation of their functionality. Furthermore, many of these tools implement their functionality in a monolithic module. This causes the GIS application developed with any of these tools to be heavily-dependent on the GIS development tool. This applications cannot easily be ported to other platforms, or modified to use a different GIS development tool.
- *Efficiency problems.* The architecture of GIS applications created using GIS development tools does not follow the guidelines given in our proposal. This causes many efficiency problems.
- *Missing functionality.* Some of the functionality described in the requirements enumerated in Section 4.2 is missing in many GIS development tools (e.g., cartographic layers, topological relationships). This functionality has to be implemented in an *ad hoc* manner in the GIS applications.

We are currently working on the implementation of a GIS development tool that follows our architecture proposal and uses open-source components[1].

[1]This work is being granted by the Xunta de Galicia (ref. PGIDIT02SIN10501PR).

Chapter 7

Conclusions and Future Work

7.1 Summary

Geographic information is slowly becoming an important element in computer systems. Many applications are being developed for many industrial, administrative and research tasks in which geographic information is the central component. Furthermore, geographic information is providing added-value to many applications that did not consider it before (e.g., location-based services). However, geographic information is a special kind of information that cannot be represented, manipulated and visualized using the methods that were traditionally used for other business and scientific information. Geographic information requires special modeling and analysis methods. The first contribution of our work is an exhaustive and detailed analysis of the special characteristics of geographic information, and the additional requirements that these characteristics impose on the architecture of geographic information systems with respect to the architecture of traditional general-purpose information systems.

The OpenGIS Consortium and the ISO Technical Committee 211 are working on a set of standard specifications for the representation, manipulation, and visualization of geographic information that will allow in the future to define the complete architecture of a GIS application using standards. However, their progress is slow and their task is not yet finished.

The main contribution of our work is a proposal for a generic architecture for geographic information systems whose design is based on these two fundamental elements:

- The analysis of the special characteristics that make geographic information systems different from traditional general-purpose information systems.

- The existing proposals of the OGC and the ISO/TC 211 of standard models, services, and languages for geographic information systems.

Our architecture proposal conforms to the OGC and ISO specifications wherever standards have been published, and provides additional detail wherever specifications are missing. In this work, we have fully described their proposals, how our architecture takes them into account, and the additional features enabled by our proposal. Moreover, the design of the architecture takes into account the special nature and characteristics of geographic information and, at the same time, the well-known requirements for general-purpose information systems.

In order to validate the architecture proposal, we have applied the architecture in the development of a complex GIS application for the Provincial Council of A Coruña. This is the third major contribution of our work. We have presented the GIS applications developed for the EIEL project, and we have compared the architecture of these applications to our proposal. Commercial GIS development tools do not satisfy completely the requirements derived from the special characteristics of geographic information and the requirements of general-purpose information systems, which caused differences between the proposed architecture and the GIS applications developed. The use of commercial GIS development tools in the analysis, design and implementation of GIS applications causes efficiency problems and lack of functionality in some cases. Nevertheless, new commercial tools such as Oracle Spatial Option or ESRI Spatial Data Extender are moving towards the right direction by implementing international standards for representing geographic information, and providing a well-structured solution for the functionality they provide.

We have also shown in our architecture proposal that there is much functionality that is independent of the particular application of the GIS. This functionality can be implemented in a generic manner, and adapted later on with application-specific details given using high-level languages. However, current commercial GIS development tools do not provide this functionality, and therefore it must be implemented *ad hoc* using programming languages.

7.2 Future Work

The next steps suggested by our work are:

- *Migrate the EIEL GIS to Oracle Spatial Option.* Oracle Spatial Option provides a correct implementation of the functionality proposed for our architecture in the sense that the representation and manipulation of geographic information is performed within the DBMS. We are currently migrating the EIEL GIS to Oracle Spatial Option, and we want to measure the improvement in performance resulting from this migration,

- *Implement the generic modules of the architecture.* Even though we have already implemented some of the generic modules described in the architecture for the EIEL project described in Chapter 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 and building modules independent of the supporting technology.

- *Develop tools to create geographic information systems based on the architecture.* Once the generic modules of the architecture are implemented, it is possible to build specific GIS applications by integrating the appropriate modules. It is also possible and desirable to define high-level languages and visual tools that allow a developer to easily select and integrate the modules for a specific GIS application.

Bibliography

[ABF$^+$02] C. Amil, N.R. Brisaboa, A. Fariña Martínez, M.R. Luaces, M.R. Penabad, A.S. Places, and J.R. Viqueira. Una Interfaz Web para un Sistema Geográfico de Información Turística. In *Actas de la II Jornada de Sistemas de Información Geográfica (JSIG)*, El Escorial, Spain, 2002.

[ABF$^+$03] C. Amil, N.R. Brisaboa, A. Fariña Martínez, M.R. Luaces, M.R. Penabad, A.S. Places, and J.R. Viqueira. Using Geographical Information Systems to Browse Touristic Information. *Information Technology & Tourism*, 6(1), 2003.

[Adl01] D.W. Adler. IBM DB2 Spatial Extender - Spatial data within the RDBMS. In *Proc. of the 27th International Conference on Very Large Data Bases(VLDB '01)*, pp. 687–692, Orlando, 2001.

[Ban88] F. Bancilhon. Object-Oriented Database Systems. In *Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 152–162, Austin, Texas, 1988.

[BCF$^+$01a] N.R. Brisaboa, J.A. Cotelo Lema, A. Fariña Martinez, M.R. Luaces, and J.R. Viqueira. E.I.E.L.: Una Experiencia de un Desarrollo SIG. In *Actas de la I Jornada de Sistemas de Información Geográfica (JSIG)*, Almagro, Spain, 2001.

[BCF$^+$01b] N.R. Brisaboa, J.A. Cotelo Lema, A. Fariña Martinez, M.R. Luaces, and J.R. Viqueira. S.I.T.P.A.C.: A Territorial Information System for A Coruña Province. In *Proc. of the 8th International Congress on Computer Science Research (CIICC)*, Colima, Mexico, 2001.

[BCF$^+$03] N.R. Brisaboa, J.A. Cotelo Lema, A. Fariña Martínez, M.R. Luaces, and J.R. Viqueira. The E.I.E.L. Project: An Experience of GIS Development. In *Proc. of the 9th EC-GI & GIS Workshop (ECGIS)*, A Coruña, Spain, 2003.

[BCLV01] N.R. Brisaboa, J.A. Cotelo Lema, M.R. Luaces, and J.R. Viqueira. State of the Art and Requirements in GIS. In *Proc. of the 3rd Mexican International Conference on Computer Science (ENC)*, Aguascalientes, Mexico, 2001.

[BCLV02] N.R. Brisaboa, J.A. Cotelo Lema, M.R. Luaces, and J.R. Viqueira. S.I.T.P.A.C.: A Territorial Information System for A Coruña Province. *I+D Computación*, 1(2), 2002.

[BCLV03] N.R. Brisaboa, J.A. Cotelo Lema, M.R. Luaces, and J.R. Viqueira. Sistemas de Información Geográfica: Revisión de su Estado Actual. In N.R. Brisaboa, ed., *Ingeniería del Software en la Década del 2000*, pp. 77–94. Tórculo, A Coruña, Spain, 2003.

[BM98] P. Burrough and R. McDonnell. *Principles of Geographical Information Systems*. Oxford University Press, 1998. ISBN: 0-19-823365-5.

[BRJ98] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, Reading, 1 Edition, 1998.

[CFG$^+$03] J.A. Cotelo Lema, L. Forlizzi, R.H. Güting, E. Nardelli, and M. Schneider. Algorithms for Moving Objects Databases. *The Computer Journal*, 46(6):680–712, 2003.

[CG02] J.A. Cotelo Lema and R.H. Güting. Dual Grid: A New Approach for Robust Spatial Algebra Implementation. *GeoInformatica*, 6(1):57–76, 2002.

[Che76] P.P.S.S. Chen. The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[Chr75] N.R. Chrisman. Topological Information Systems for Geographic Representation. In *Proc. of the Second International Symposium on Computer-Assisted Cartography (Auto-Carto 2)*, pp. 346–351, Falls Church, 1975.

[Chr78] N.R. Chrisman. Concepts Of Space as a Guide to Cartographic Data Structures. In *Proc. of the First International Advanced Study Symposium on Topological Data Structures for Geographic Information Systems*, pp. 1–19, Cambridge, Massachusetts, 1978.

[Cod70] E.F. Codd. A Relational Model for Large Shared Data Banks. *Communications of the ACM*, 13(6):377–387, 1970.

[Cot01] J.A. Cotelo Lema. An Analysis of Consistency Properties in Existing Spatial and Spatiotemporal Data Models. In *Proc. of the 5thEast-European Conf. on Advances in Databases and Information Systems (ADBIS)*, Vilnius, Lithuania, 2001.

[Cou92] H. Couclelis. People Manipulate Objects (But Cultivate Fields): Beyond the Raster-Vector Debate in GIS. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space: Proc. of the International Conference GIS*, pp. 65–77, Berlin, Heidelberg, 1992.

[Dav98] J. Davis. IBM's DB3 Spatial Extender: Managing Geo-Spatial Information within the DBMS. Technical report, IBM, 1998.

[DDL03] C.J. Date, H. Darwen, and N. Lorentzos. *Temporal Data and the Relational Model*. Morgan Kaufmann Publishers, 2003. ISBN: 1-55860-855-9.

[DG00a] S. Dieker and R.H. Guting. Plug and Play with Query Algebras: SECONDO-A Generic DBMS Development Environment. In *Proc. of the International Database Engineering and Application Symposium (IDEAS)*, pp. 380–392, 2000.

[DG00b] S. Dieker and R.H. Guting. Efficient Handling of Tuples with Embedded Large Objects. *Data Knowledge Engineering*, 32(3):247–269, 2000.

[DGL00] S. Dieker, R.H. Güting, and M.R. Luaces. A Tool for Nesting and Clustering Large Objects. In *Proc. of the 12th Intl. Conf. on Scientific and Statistical Database Management (SSDBM)*, pp. 169–181, Berlin, Germany, 2000.

[Die01] S. Dieker. *Efficient Integration of Query Algebra Modules into an Extensible Database Framework*. Mensch und Buch Verlag, Berlin, 2001. ISBN 3-89820-226-7.

[dOGM99] J.L. de Oliveira, M.A. Goncalves, and C.B. Medeiros. A Framework for Designing and Implementing the User Interface of a Geographic Digital Library. *Int. J. on Digital Libraries*, 2(2-3):190–206, 1999.

[EF88] M.J. Egenhofer and A.U. Frank. Towards a Spatial Query Language: User Interface Considerations. In *Proceedings of the 14th Conference on Very Large Databases, Morgan Kaufman pubs. (Los Altos CA), Bancilhon and DeWitt (eds), Los Angeles*, 1988.

[Ege90] M.J. Egenhofer. Interaction with Geographic Information Systems via Spatial Queries. *Journal of Visual Languages and Computing*, 1:389–413, 1990.

[Fra03] A.U. Frank. Ontology for Spatio-temporal Databases. In M. Koubarakis, T.K. Sellis, A.U. Frank, S. Grumbach, R.H. Güting, C.S. Jensen, N.A. Lorentzos, Y. Manolopoulos, E. Nardelli, B. Pernici, H. Schek, M. Scholl, B. Theodoulidis, and N.Tryfona, eds., *Spatio-Temporal Databases:*

The CHOROCHRONOS Approach, volume 2520 of *Lecture Notes in Computer Science*, pp. 9–77. Springer, 2003.

[GBE+00] R.H. G&#252;ting, M.H. Böhlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM Trans. Database Syst.*, 25(1):1–42, 2000.

[GDF+99] R.H. Güting, S. Dieker, C. Freundorfer, L. Becker, and H. Schenk. SECONDO/QP: Implementation of a Generic Query Processor. In *Proc. of the 10th Intl. Conf. on Database and Expert Systems Applications (DEXA)*, pp. 66–87, 1999.

[GG98] V. Gaede and O. Günther. Multidimensional access methods. *ACM Comput. Surv.*, 30(2):170–231, 1998.

[Güt94] R.H. Güting. An Introduction to Spatial Database Systems. *VLDB Journal*, 3(4):357–400, 1994.

[HA03] J.E. Harmon and S.J. Anderson. *The Design and Implementation of Geographic Information Systems*. John Wiley & Sons, 2003. ISBN: 0-471-20488-9.

[ISO00a] Geographic Information – Conformance and Testing. International Standard 19105, ISO/IEC, 2000.

[ISO00b] Geographic Information – Imagery and Gridded Data. International standard 19121, ISO/IEC, 2000.

[ISO00c] Geographic Information – Simple Feature Access - Part 1: Common Architecture. Draft International standard 19125-1, ISO/IEC, 2000.

[ISO00d] Geographic Information – Simple Feature Access - Part 2: SQL Option. Draft International standard 19125-2, ISO/IEC, 2000.

[ISO01a] Geographic Information – Methodology for Feature Cataloguing. Draft International standard 19110, ISO/IEC, 2001.

[ISO01b] Geographic Information – Functional Standards. International standard 19120, ISO/IEC, 2001.

[ISO02a] Geographic Information – Reference Model. International Standard 19101, ISO/IEC, 2002.

[ISO02b] Geographic Information – Terminology. Draft Technical Specification 19104, ISO/IEC, 2002.

[ISO02c] Geographic Information – Profiles. Draft Technical Specification 19106, ISO/IEC, 2002.

[ISO02d] Geographic Information – Temporal Schema. International standard 19108, ISO/IEC, 2002.

[ISO02e] Geographic Information – Rules for Application Schema. Draft International Standard 19109, ISO/IEC, 2002.

[ISO02f] Geographic Information – Quality Principles. International standard 19113, ISO/IEC, 2002.

[ISO02g] Geographic Information – Positioning Services. Draft International standard 19116, ISO/IEC, 2002.

[ISO02h] Geographic Information – Portrayal. Draft International standard 19117, ISO/IEC, 2002.

[ISO02i] Geographic Information – Encoding. Draft International standard 19118, ISO/IEC, 2002.

[ISO02j] Geographic Information – Services. Draft International standard 19117, ISO/IEC, 2002.

[ISO02k] Geographic Information – Schema for coverage geometry and functions. Draft International standard 19123, ISO/IEC, 2002.

[ISO03a] Geographic Information – Conceptual Schema Language. Draft Technical Specification 19103, ISO/IEC, 2003.

[ISO03b] Geographic Information – Spatial Schema. International standard 19107, ISO/IEC, 2003.

[ISO03c] Geographic Information – Spatial Referencing by Coordinates. International standard 19111, ISO/IEC, 2003.

[ISO03d] Geographic Information – Spatial Referencing by Geographic Identifiers. International standard 19112, ISO/IEC, 2003.

[ISO03e] Geographic Information – Metadata. International standard 19115, ISO/IEC, 2003.

[ISO04a] Geographic Information – Web Map Server Interface. Draft International standard 19128, ISO/IEC, 2004.

[ISO04b] Geographic Information – Location Based Services Tracking and Navigation. Draft International standard 19133, ISO/IEC, 2004.

[ISO32] Geographic Information – Quality Evaluation Procedures. International standard 19114, ISO/IEC, 2032.

[KSF$^+$03] M. Koubarakis, T.K. Sellis, A.U. Frank, S. Grumbach, R.H. Güting, C.S. Jensen, N.A. Lorentzos, Y. Manolopoulos, E. Nardelli, B. Pernici, H. Schek, M. Scholl, B. Theodoulidis, and N.Tryfona, eds. *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, volume 2520 of *Lecture Notes in Computer Science*. Springer, 2003.

[Kuh91] W. Kuhn. Are Displays Maps or Views? In *Proc. of the AutoCarto 10*, pp. 261–274, Baltimore, 1991.

[LBPV04a] M.R. Luaces, N.R. Brisaboa, J.R. Paramá, and J.R. Viqueira. A Generic Architecture for Geographic Information Systems. In *Proc. of the 2nd International Symposium on Innovation in Information & Communication Technology (ISIICT)*, Amman, Jordan, 2004.

[LBPV04b] M.R. Luaces, N.R. Brisaboa, J.R. Paramá, and J.R. Viqueira. A Generic Framework for Geographic Information Systems. Submitted to the 4th International Workshop on Web and Wireless Geographical Information Systems (W^{2}GIS) Koyang, Korea, 2004.

[LGMR01] P. Longley, M. Goodchild, D. Maguire, and D. Rhind. *Geographic Information Systems and Science*. John Wiley & Sons, 2001. ISBN: 0-471-49521-2.

[LT92] R. Laurini and D. Thompson. *Fundamentals of spatial informations systems*. The APIC Series Nř 37 - Academic Press, 1992. ISBN: 0-12-438380-7.

[Lua00] M.R. Luaces. A Spatio-Temporal Algebra Implementation. In *Proc. of the 5th World Conference on Integrated Design and Process Technology (IDPT)*, Dallas, USA, 2000.

[Lua01] M.R. Luaces. Una Interfaz de Usuario para Datos Espacio-temporales. In M. Toro, J.G. Consuegra, and M. Piattini, eds., *Sistemas de Información Geográficos. Una aproximación desde la Ingeniería del Software y las Bases de Datos*, pp. 191–196. Fundación DINTEL para la Difusión de las Ingenierías Informática y de Telecomunicaciones, Madrid, Spain, 2001.

[OCG01a] OpenGIS Implementation Specification: Coordinate Transformation Services. OpenGIS Project Document 01-009, Open GIS Consortium, Inc., 2001.

[OCG01b] Filter Encoding Implementation Specification. OpenGIS Project Document 02-059, Open GIS Consortium, Inc., 2001.

[OCG02a] OpenGIS Styled Layer Descriptor Implementation Specification. OpenGIS Project Document 02-070, Open GIS Consortium, Inc., 2002.

[OCG02b] OpenGIS Web Feature Service Implementation Specification. OpenGIS Project Document 02-058, Open GIS Consortium, Inc., 2002.

[OCG02c] OpenGIS Web Map Service Implementation Specification. OpenGIS Project Document 01-068r3, Open GIS Consortium, Inc., 2002.

[OCG03a] OpenGIS Geography Markup Language (GML) Implementation Specification, Version 3.00. OpenGIS Project Document 02-023r4, Open GIS Consortium, Inc., 2003.

[OCG03b] OpenGIS Web Coverage Service Implementation Specification. OpenGIS Project Document 03-065r6, Open GIS Consortium, Inc., 2003.

[OGC98] OpenGIS Simple Features Specification For CORBA. Revision 1.0. OpenGIS Project Document 99-054, Open GIS Consortium, Inc., 1998.

[OGC99a] OpenGIS Simple Features Specification For OLE/COM. Revision 1.1. OpenGIS Project Document 99-050, Open GIS Consortium, Inc., 1999.

[OGC99b] OpenGIS Simple Features Specification For SQL. Revision 1.1. OpenGIS Project Document 99-049, Open GIS Consortium, Inc., 1999.

[OGC99c] OpenGIS Abstract Specification. Topic 0: Abstract Specification Overview. OpenGIS Project Document 99-100r1, Open GIS Consortium, Inc., 1999.

[OGC99d] OpenGIS Abstract Specification. Topic 10: Feature Collections. OpenGIS Project Document 99-110, Open GIS Consortium, Inc., 1999.

[OGC99e] OpenGIS Abstract Specification. Topic 13: Catalog Services. OpenGIS Project Document 99-113, Open GIS Consortium, Inc., 1999.

[OGC99f] OpenGIS Abstract Specification. Topic 14: Semantics and Information Communities. OpenGIS Project Document 99-114, Open GIS Consortium, Inc., 1999.

[OGC99g] OpenGIS Abstract Specification. Topic 4: Locational Geometry Structures. OpenGIS Project Document 99-103, Open GIS Consortium, Inc., 1999.

[OGC99h] OpenGIS Abstract Specification. Topic 4: Stored Functions and Interpolation. OpenGIS Project Document 99-104, Open GIS Consortium, Inc., 1999.

[OGC99i] OpenGIS Abstract Specification. Topic 5: Features. OpenGIS Project Document 99-105r2, Open GIS Consortium, Inc., 1999.

[OGC99j] OpenGIS Abstract Specification. Topic 7: Earth Imagery. OpenGIS Project Document 99-107, Open GIS Consortium, Inc., 1999.

[OGC99k] OpenGIS Abstract Specification. Topic 8. Relationships Between Features. OpenGIS Project Document 99-108r2, Open GIS Consortium, Inc., 1999.

[OGC99l] OpenGIS Abstract Specification. Topic 9: Quality. OpenGIS Project Document 99-109r1, Open GIS Consortium, Inc., 1999.

[OGC00a] OpenGIS Abstract Specification. Topic 15: Image Exploitation Services. OpenGIS Project Document 00-115, Open GIS Consortium, Inc., 2000.

[OGC00b] OpenGIS Abstract Specification. Topic 16: Image Coordinate Transformation Services. OpenGIS Project Document 00-116, Open GIS Consortium, Inc., 2000.

[OGC00c] OpenGIS Abstract Specification. Topic 6: The Coverage Type. OpenGIS Project Document 00-106, Open GIS Consortium, Inc., 2000.

[OGC01a] OpenGIS Abstract Specification. Topic 1: Feature Geometry. OpenGIS Project Document 01-101, Open GIS Consortium, Inc., 2001.

[OGC01b] OpenGIS Abstract Specification. Topic 11: OpenGIS Metadata. OpenGIS Project Document 01-111, Open GIS Consortium, Inc., 2001.

[OGC02a] OpenGIS Catalog Services Specification. OpenGIS Project Document 02-087r3, Open GIS Consortium, Inc., 2002.

[OGC02b] OpenGIS Abstract Specification. Topic 12: OpenGIS Service Architecture. OpenGIS Project Document 02-112, Open GIS Consortium, Inc., 2002.

[OGC02c] I. Open GIS Consortium. OpenGIS Geography Markup Language (GML) Implementation Specification, Version 2.1.2. OpenGIS Project Document 02-069, Open GIS Consortium, Inc., 2002.

[OGC03a] OpenGIS Reference Model. OpenGIS Project Document 03-040, Open GIS Consortium, Inc., 2003.

[OGC03b] OpenGIS Abstract Specification. Topic 2: Spatial Referencing by Coordinates. OpenGIS Project Document 03-073r1, Open GIS Consortium, Inc., 2003.

[PR95] J.P. Peloux and P. Rigaux. A Loosely Coupled Interface to an Object-Oriented Geographic Database. In *Spatial Information Theory*, pp. 123–137, 1995.

[PS85] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, New York, 1985.

[RBP$^+$91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall International Editions, New York, NY, 1991.

[RS94] P. Rigaux and M. Scholl. Multiple Representation Modelling and Querying. In *IGIS*, pp. 59–69, 1994.

[RS95] P. Rigaux and M. Scholl. Multi-Scale Partitions: Application to Spatial and Statistical Databases. In *Symposium on Large Spatial Databases*, pp. 170–183, 1995.

[RSV93] P. Rigaux, M. Scholl, and A. Voisard. A Map Editing Kernel Implementation: Application to Multiple Scale Display. *Lecture Notes in Computer Science*, 716:341–??, 1993.

[RSV01] P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases With Application To GIS.* Academic Press, 2001. ISBN: 1-55680-588-6.

[Sam90] H. Samet. *The design and analysis of spatial data structures.* Addison-Wesley Longman Publishing Co., Inc., 1990.

[SCR$^+$99] S. Shekhar, S. Chawla, S. Ravada, A. Fetterer, X. Liu, and C. tien Lu. Spatial Databases-Accomplishments and Research Needs. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):45–55, 1999.

[SM96] M. Stonebraker and D. Moore. *Object-Relational DBMSs: The Next Great Wave.* Morgan Kaufmann Publ., San Francisco, 1996.

[Tom90] C. Tomlin. *Geographic Information Systems and Cartographic Modelling.* Englewood Cliffs, 1990.

[TVM00] T. Tzouramanis, M. Vassilakopoulos, and Y. Manolopoulos. Overlapping Linear Quadtrees and Spatio-Temporal Query Processing. *The Computer Journal,*, 43(4):325–343, 2000.

[Viq03] J.R. Viqueira. *Formal Extension of the Relational Model for the Management of Spatial and Spatio-temporal Data.* PhD thesis, Universidade da Coruña, Spain, 2003.

[Voi94] A. Voisard. Designing and Integrating User Interfaces of Geographic Database Applications. In *Advanced Visual Interfaces*, pp. 133–142, 1994.

[Voi95] A. Voisard. Mapgets: A Tool for Visualizing and Querying Geographic Information. *Journal of Visual Languages and Computing*, 6, 1995.

Appendix A

International Standards

The following list enumerates the Abstract Specification Topics published by the OpenGIS Consortium:

Topic 1 Feature Geometry [OGC01a]
Topic 2 Spatial Referencing by Coordinates [OGC03b]
Topic 3 Locational Geometry Structures [OGC99g]
Topic 4 Stored Functions and Interpolation [OGC99h]
Topic 5 Features [OGC99i]
Topic 6 The Coverage Type [OGC00c]
Topic 7 Earth Imagery [OGC99j]
Topic 8 Relationships Between Features [OGC99k]
Topic 9 Quality [OGC99l]
Topic 10 Feature Collections [OGC99d]
Topic 11 Metadata [OGC01b]
Topic 12 The OpenGIS Service Architecture [OGC02b]
Topic 13 Catalog Services [OGC99e]
Topic 14 Semantics and Information Communities [OGC99f]
Topic 15 Image Exploitation Services [OGC00a]
Topic 16 Image Coordinate Transformation Services [OGC00b]

Now, we enumerate the International Standards published by the ISO/TC 211:

ISO/IS 19101 Geographic information – Reference model [ISO02a]
ISO/IS 19105 Geographic information – Conformance and testing [ISO00a]
ISO/IS 19107 Geographic information – Spatial schema [ISO03b]
ISO/IS 19108 Geographic information – Temporal schema [ISO02d]
ISO/IS 19111 Geographic information – Spatial referencing by coordinates [ISO03c]

ISO/IS 19112 Geographic information – Spatial referencing by geographic identifiers [ISO03d]
ISO/IS 19113 Geographic information – Quality principles [ISO02f]
ISO/IS 19114 Geographic information – Quality evaluation procedures [ISO32]
ISO/IS 19115 Geographic information – Metadata [ISO03e]
ISO/IS 19120 Geographic information – Functional standards [ISO01b]
ISO/IS 19121 Geographic information – Imagery and gridded data [ISO00b]

In addition to approved and published international standards, we enumerate the ISO/TC 211 International Standards that are still at a draft stage:

ISO/DIS 19104 Geographic information – Terminology [ISO02b]
ISO/DIS 19106 Geographic information – Profiles [ISO02c]
ISO/DIS 19109 Geographic information – Rules for application schema [ISO02e]
ISO/DIS 19110 Geographic information – Feature cataloguing methodology [ISO01a]
ISO/DIS 19116 Geographic information – Positioning services [ISO02g]
ISO/DIS 19117 Geographic information – Portrayal [ISO02h]
ISO/DIS 19118 Geographic information – Encoding [ISO02i]
ISO/DIS 19119 Geographic information – Services [ISO02j]
ISO/DIS 19123 Geographic information – Schema for coverage geometry and functions [ISO02k]
ISO/DIS 19125-1 Geographic information – Simple feature access – Part 1: Common architecture [ISO00c]
ISO/DIS 19125-2 Geographic information – Simple feature access – Part 2: SQL option [ISO00d]
ISO/DIS 19128 Geographic information – Web Map Server interface [ISO04a]
ISO/DIS 19133 Geographic information – Location based services tracking and navigation [ISO04b]

Finally, the following list shows the implementation specifications published by the OpenGIS Consortium:

CT Coordinate Transformation Services [OCG01a]
Filter Filter Encoding [OCG01b]
GML Geography Markup Language [OCG03a]
SFC Simple Features – CORBA [OGC98]
SFS Simple Features – SQL [OGC99b]
SFO Simple Features – OLE/COM [OGC99a]
SLD Styled Layer Descriptor [OCG02a]
WCS Web Coverage Service [OCG03b]
WFS Web Feature Service [OCG02b]
WMS Web Map Service [OCG02c]

Appendix B

Publications and Other Research Results Related to the Thesis

B.1 Publications

B.1.1 International Conferences

- M.R. Luaces, N.R. Brisaboa, J.R. Paramá, and J.R. Viqueira. A Generic Framework for Geographic Information Systems. Submitted to the 4th International Workshop on Web and Wireless Geographical Information Systems (W^2GIS) Koyang, Korea, 2004.

- M.R. Luaces, N.R. Brisaboa, J.R. Paramá, and J.R. Viqueira. A Generic Architecture for Geographic Information Systems. In *Proc. of the 2nd International Symposium on Innovation in Information & Communication Technology (ISIICT)*, Amman, Jordan, 2004.

- N.R. Brisaboa, J.A. Cotelo Lema, A. Fariña Martínez, M.R. Luaces, and J.R. Viqueira. The E.I.E.L. Project: An Experience of GIS Development. In *Proc. of the 9th EC-GI & GIS Workshop (ECGIS)*, A Coruña, Spain, 2003.

- N.R. Brisaboa, J.A. Cotelo Lema, A. Fariña Martinez, M.R. Luaces, and J.R. Viqueira. S.I.T.P.A.C.: A Territorial Information System for A Coruña Province. In *Proc. of the 8th International Congress on Computer Science Research (CIICC)*, Colima, Mexico, 2001.

- N.R. Brisaboa, J.A. Cotelo Lema, M.R. Luaces, and J.R. Viqueira. State of the Art and Requirements in GIS. In *Proc. of the 3rd Mexican International Conference on Computer Science (ENC)*, Aguascalientes, Mexico, 2001.
- S. Dieker, R.H. Güting, and M.R. Luaces. A Tool for Nesting and Clustering Large Objects. In *Proc. of the 12th Intl. Conf. on Scientific and Statistical Database Management (SSDBM)*, pp. 169–181, Berlin, Germany, 2000.
- M.R. Luaces. A Spatio-Temporal Algebra Implementation. In *Proc. of the 5th World Conference on Integrated Design and Process Technology (IDPT)*, Dallas, USA, 2000.

B.1.2 National Conferences

- C. Amil, N.R. Brisaboa, A. Fariña Martínez, M.R. Luaces, M.R. Penabad, A.S. Places, and J.R. Viqueira. Una Interfaz Web para un Sistema Geográfico de Información Turística. In *Actas de la II Jornada de Sistemas de Información Geográfica (JSIG)*, El Escorial, Spain, 2002.
- N.R. Brisaboa, J.A. Cotelo Lema, A. Fariña Martinez, M.R. Luaces, and J.R. Viqueira. E.I.E.L.: Una Experiencia de un Desarrollo SIG. In *Actas de la I Jornada de Sistemas de Información Geográfica (JSIG)*, Almagro, Spain, 2001.

B.1.3 Journals and Book Chapters

- N.R. Brisaboa, J.A. Cotelo Lema, M.R. Luaces, and J.R. Viqueira. Sistemas de Información Geográfica: Revisión de su Estado Actual. In N.R. Brisaboa, ed., *Ingeniería del Software en la Década del 2000*, pp. 77–94. A Coruña, Spain, 2003.
- C. Amil, N.R. Brisaboa, A. Fariña Martínez, M.R. Luaces, M.R. Penabad, A.S. Places, and J.R. Viqueira. Using Geographical Information Systems to browse touristic information. *Information Technology & Tourism*, 6(1), 2003.
- N.R. Brisaboa, J.A. Cotelo Lema, M.R. Luaces, and J.R. Viqueira. S.I.T.P.A.C.: A Territorial Information System for A Coruña Province. *I+D Computación*, 1(2), 2002.
- M.R. Luaces. Una Interfaz de Usuario para Datos Espacio-temporales. In M. Toro, J.G. Consuegra, and M. Piattini, eds., *Sistemas de Información Geográficos. Una aproximación desde la Ingeniería del Software y las Bases de Datos*, pp. 191–196. Fundación DINTEL para la Difusión de las Ingenierías Informática y de Telecomunicaciones, Madrid, Spain, 2001.

B.2 Research Stays

- *August 1st, 1998 - July 31st, 2000.* Working as a member of the CHOROCHRONOS project at the FernUniversität Hagen in Hagen, Germany, under the supervision of Prof. Dr. Ralf Hartmut Güting.
- *December 3rd, 2001 - December 23rd, 2001.* Research stay at the FernUniversität Hagen in Hagen, Germany, under the supervision of Prof. Dr. Ralf Hartmut Güting.
- *May 21st, 2003 - July 23rd, 2003.* Research stay at the FernUniversität Hagen in Hagen, Germany, under the supervision of Prof. Dr. Ralf Hartmut Güting.

Appendix C

Descripción del Trabajo Presentado

C.1 Introducción

La información geográfica, en forma de mapas en papel, ha sido uno de los motores del progreso de nuestra sociedad durante muchos siglos. Hasta hace unas pocas décadas, manipular, sintetizar y representar información geográfica estaba restringido a mapas en papel y estas tareas estaban limitadas a procesos manuales y sin interactividad. La mejora exponencial en la capacidad de las tecnologías basadas en ordenador, y la demanda creciente para manipular y analizar información geográfica de forma interactiva han creado la necesidad de *sistemas de información geográfica* (SIG).

Se han propuesto muchas definiciones para el término SIG en la literatura del tema, cada una considerando la funcionalidad del sistema desde una perspectiva diferente. Todas ellas se centran en tres aspectos diferentes:

1. un SIG es una *colección de herramientas informáticas para realizar análisis y simulaciones de tipo geográfico* [LT92, BM98, RSV01],

2. un SIG se apoya en un conjunto de *estructuras de datos y algoritmos para representar, recuperar y manipular información geográfica* [Wor95, RSV01],

3. y un SIG es una utilidad que *ayuda a la gente a tomar decisiones en tareas relacionadas con la geografía* [LGMR01, HA03].

Podemos extraer nuestra propia definición de éstas diciendo que un SIG es un sistema informático que permite *modelar, capturar, almacenar, manipular, consultar,* *recuperar, analizar y e visualizar eficientemente información, siendo parte de la información de naturaleza geográfica.*

Una característica importante de los sistemas de información geográfica es que son más que herramientas para producir mapas en papel. Mientras que en la cartografía tradicional el mapa en papel es la base de datos, en un SIG el mapa es sólo una proyección de una vista particular de la base de datos geográfica en un instante determinado. Esto permite que el usuario final del SIG pueda revisar un número ilimitado de alternativas de análisis, y que pueda realizar mapas desde diferentes puntos de vista enfatizando diferentes aspectos de la información [BM98]. Como consecuencia de esto, los requisitos funcionales para SIG son muy amplios y van más allá de los requisitos para sistemas de información tradicionales. Por ejemplo, aplicaciones para la administración pública y el análisis de mercados requieren integrar y mostrar información con diferentes niveles de resolución, aplicaciones de control de flotas necesitan operaciones de búsqueda de rutas en redes de transporte, y sistemas de información medioambientales que tratan con el análisis del terreno necesitan interpolación de datos.

Tras muchos años de investigación y desarrollo, hoy en día se acepta que la arquitectura de un sistema de información de propósito general debe estar formada por tres capas separadas: la *capa de presentación*, la *capa de lógica de la aplicación*, y la *capa de datos* (ver la Figura C.1). La ventaja principal de esta arquitectura es que asegura la separación estricta de la funcionalidad del sistema en tres módulos independientes que interactúan sólo en las interfaces. Esto permite al desarrollador modificar cualquiera de los módulos de la aplicación con un impacto mínimo en los otros. Por lo tanto, esta arquitectura permite un incremento de la flexibilidad, mantenibilidad, reusabilidad y escalabilidad. Por ejemplo, si el modelo de datos relacional se usa en la capa de datos de un sistema de información de propósito general, es posible utilizar cualquier sistema gestor de bases de datos relacional (SGBD) como fuente de datos con un impacto mínimo en la aplicación.

Aún cuando la arquitectura en tres capas para los sistemas de información de propósito general es aplicable para los SIG, la naturaleza especial y las características exclusivas de la información geográfica imponen requisitos funcionales especial en la arquitectura de las aplicaciones SIG en términos de modelos conceptual y lógicos, estructuras de datos, métodos de acceso, técnicas de análisis, y procedimientos de visualización. Por ejemplo:

- *Se necesitan tipos de datos y operaciones especiales para representar y manipular información geográfica.* Los tipos de datos disponibles en los modelos lógicos para sistemas de información tradicionales son aptos para representar información típica de negocios utilizando números, textos o fechas. Sin embargo, la representación de información geográfica en un ordenador requiere nuevos tipos de datos como *punto*, *curva* o *superficie*. De forma similar, las operaciones proporcionadas por los modelos lógicos para sistemas

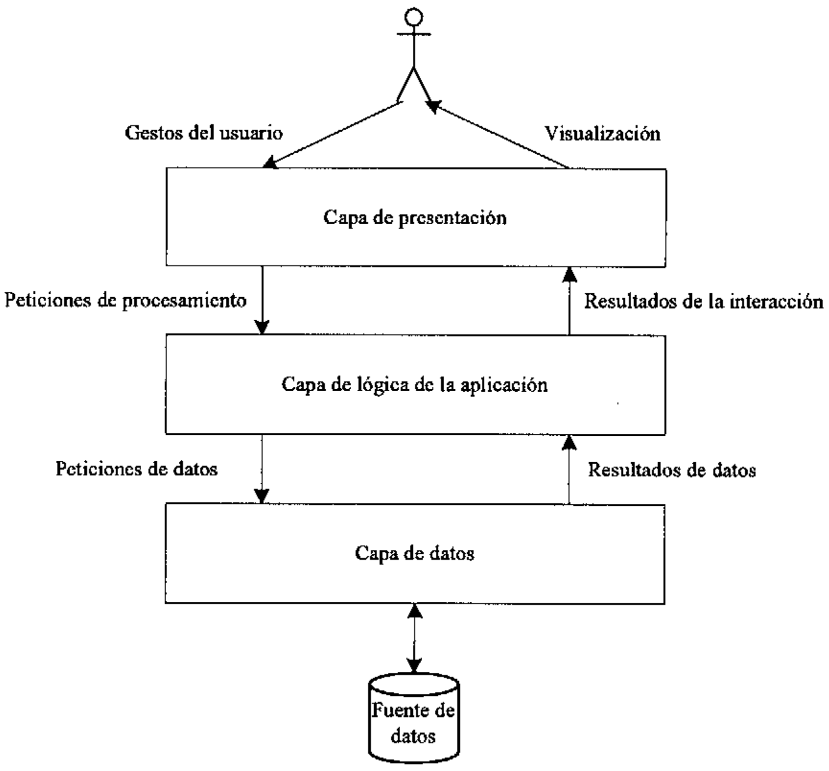


Figure C.1: Arquitectura de Sistemas de Información en Tres Capas.

de información tradicionales permiten manipular valores de tipos de datos típicos de negocios (por ejemplo, operaciones aritméticas, o procesado simple de cadenas de texto). Para manipular información geográfica se necesitan nuevas operaciones (por ejemplo, *distancia, dirección* o *intersección*).

- *La información geográfica requiere muchos procedimientos de análisis y visualización distintos.* Hay un amplio conjunto de técnicas de análisis aplicables a la información geográfica que requieren procedimientos especiales de transformación y análisis, incluyendo entre otros *algoritmos de búsqueda de caminos en espacios topológicos*, *análisis de terrenos*, o *interpolación de datos geográficos*. Además, dado que la información geográfica necesita tipos de datos y operaciones específicos para ser representada y manipulada, también necesita nuevas técnicas de visualización para mostrar valores de estos nuevos tipos de datos y resultados de estas operaciones.

  Además, la información geográfica tiene algunas peculiaridades que tienen un gran impacto en el proceso de presentación, como por ejemplo la necesidad de diferentes abstracciones para la representación y la visualización de la información. En una base de datos relacional tradicional, se usa una abstracción similar para el modelo lógico (una relación) y para la visualización de valores (una tabla). En cambio, la abstracción utilizada en el modelo lógico de un SIG para un elemento de información (por ejemplo, un punto que representa una ciudad) es diferente de la abstracción utilizada en su visualización (por ejemplo, un símbolo, un punto, o una pequeña superficie).

- *La información geográfica típicamente es voluminosa con una estructura jerárquica impuesta naturalmente.* Mientras que la representación de datos de negocio tradicionales requiere pocos bytes de almacenamiento, la representación de datos geográficos requiere una mayor cantidad de espacio de almacenamiento. Por ejemplo, para representar la frontera de un país, se necesitan miles de pares de números reales. Esto implica que estructuras de almacenamiento, estructuras de acceso, y algoritmos eficientes para las operaciones deben ser definidas para representar y manipular información geográfica. Además, la naturaleza especial del espacio geográfico impone relaciones y una estructura jerárquica entre los valores espaciales que no existe en otros dominios de datos. Por ejemplo, las diferentes carreteras están relacionadas por una relación de conexión, y los edificios de una ciudad están relacionados por la relación *estar contenido* con respecto a la manzana de la ciudad en la que están. Estas relaciones son importantes y deben ser representadas en el sistema.

- *El procesamiento de la información geográfica se caracteriza por transacciones que mucho más largas que una transacción típica de una base de datos relacional.* Esto se debe a la característica anterior. Dado que la información geográfica es voluminosa, la creación de y modificación de datos geográficos requiere mucho más tiempo que las mismas tareas para datos de negocio tradicionales. Por lo tanto, se necesitan transacciones largar que mantengan los datos bloqueados por un periodo de tiempo más largo.

- *Hay dos visiones conceptuales diferentes del espacio geográfico.* Se reconoce ampliamente que el espacio geográfico puede ser conceptualizado de dos formas distintas: utilizando modelos *basados en campos* o utilizando modelos *basados en objetos.* Los modelos basados en campos representan la información geográfica como colecciones de distribuciones espaciales donde cada distribución puede formalizarse como una función matemática desde un marco de referencia espacial a un dominio de un atributo. En estos modelos, se considera que el espacio geográfico como un elemento con existencia propia que tiene propiedades asociadas a cada localización. Por otra parte, los modelos basados en objetos representan la información geográfica como entidades discretas e identificables que consisten en una colección de atributos que describen las propiedades de un fenómeno del mundo real utilizando valores alfanuméricos y geométricos. Estos modelos consideran el espacio geográfico como un contenedor de objetos que tienen propiedades asociadas. Cada visión es conceptualmente diferente y debe ser representada en un modelo conceptual utilizando abstracciones diferentes.

- *Cada visión conceptual del espacio puede ser representada de muchas formas diferentes en un ordenador.* En primer lugar, los modelos lógicos para los sistemas de información tradicionales no proporcionan un soporte eficiente para la representación de información geográfica. Por lo tanto, se deben definir nuevos modelos lógicos, o los existentes deben ser extendidos para soportar información geográfica. En segundo lugar, cada uno de los modelos conceptuales para información geográfica (esto es, el modelo basado en campos y el modelo basado en objetos) puede ser representado utilizando un modelo lógico de muchas formas diferentes.

Estas y otras características afectan a toda la arquitectura de un SIG. Por lo tanto, es muy importante determinar los requisitos especiales de los SIG y la funcionalidad que debe ser proporcionada por los SIG más allá de los requisitos estándar y la funcionalidad de un sistema de información de propósito general. Esto nos permitirá diseñar e implementar un SIG con las características apropiadas para modelar, reunir, consultar y visualizar información geográfica.

C.2 Metodología Utilizada

El objetivo principal de nuestro trabajo es la propuesta de una arquitectura genérica para sistemas de información geográfica que proporciona soporte para las características especiales de la información geográfica. Nuestra estrategia para conseguir este objetivo consiste en dos pasos (ver la Figura C.2):

- Primero, analizamos las características de la información geográfica que imponen requisitos particulares sobre el diseño de la arquitectura de aplicaciones de SIG.
- Entonces, adaptamos la arquitectura tradicional en tres capas para sistemas de información de propósito general para tener en cuenta los requisitos extraídos en el paso anterior.

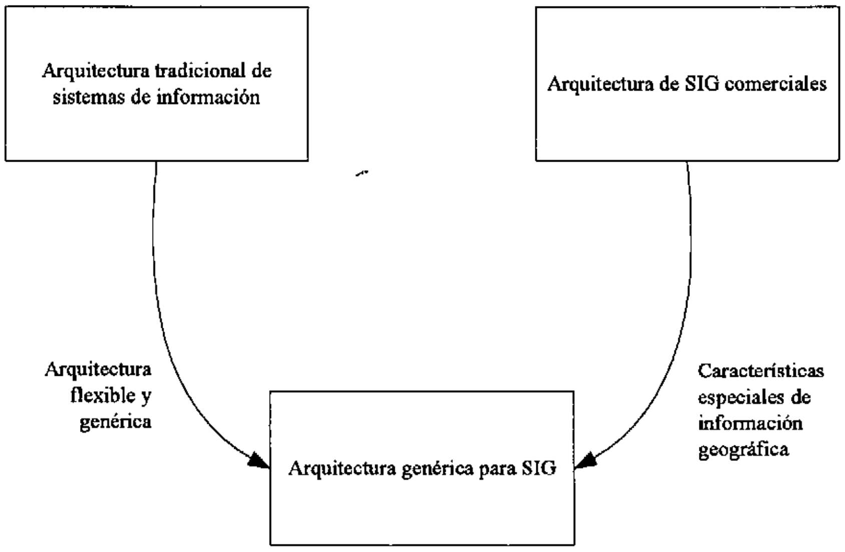

Arquitectura tradicional de sistemas de información
Arquitectura de SIG comerciales
Arquitectura flexible y genérica
Características especiales de información geográfica
Arquitectura genérica para SIG

Figure C.2: Hacia una Arquitectura Genérica para SIG.

Esta estrategia nos permite diseñar una arquitectura para SIG que mantiene los beneficios de la arquitectura tradicional en tres capas para sistemas de información de propósito general y que además proporciona soporte para los requisitos especiales de la información geográfica. Las aplicaciones SIG desarrolladas utilizando esta arquitectura heredan y disfrutan de propiedades importantes que ya están presentes en la arquitectura de sistemas de información de propósito general (por ejemplo, interoperabilidad, flexibilidad, generalidad, reusabilidad, o escalabilidad). Las propiedades más importantes de la arquitectura propuesta son su *flexibilidad* (esto es, la arquitectura puede ser adaptada para implementar aplicaciones en distintos entornos tecnológicos), y su *generalidad* (esto es, la arquitectura puede ser utilizada para múltiples aplicaciones con propósitos específicos distintos). Al mismo tiempo, estas aplicaciones de SIG tienen en cuenta los requisitos particulares de la información geográfica con respecto a la arquitectura del sistema. Para alcanzar estos objetivos, cada componente de la arquitectura debe ser descrito de forma precisa, y debe ser ilustrado con resultados de investigación, estándares existentes, o aplicaciones comerciales, según sea apropiado.

Ha habido muchos esfuerzos de investigación y desarrollo para estandarizar muchos aspectos de la tecnología de SIG, particularmente realizados por el OpenGIS Consortium (OGC) y el ISO Technical Committee 211 (*ISO/TC 211, Geographic Information/Geomatics*). Su objetivo a largo plazo es la integración completa de los datos geoespaciales y los recursos de procesamiento geográfico en la rama principal de la informática y extender el uso de software de procesamiento geográfico interoperable y de datos geográficos a lo ancho de la infraestructura de la información [OGC99c, ISO02a]. Estos objetivos se solapan con el objetivo principal de nuestro trabajo, y por lo tanto es indispensable que revisemos exhaustivamente las propuestas del OGC y de ISO. En esta revisión, mostramos las virtudes de los estándares y las especificaciones de implementación propuestas por estas organizaciones, y describimos y analizamos también las desventajas de estas propuestas. Finalmente, describimos las soluciones adoptadas en nuestra propuesta para superar estas desventajas mientras la arquitectura se mantiene conforme a los estándares disponibles.

Hoy en día existen muchas herramientas comerciales para el desarrollo de SIG que se centran en diferentes aspectos de la información geográfica y las tecnologías informáticas. Estas herramientas de desarrollo tienen limitaciones que no permiten implementar aplicaciones de SIG bien estructuradas que respondan a los requisitos especiales de la información geográfica, y que proporcionen importantes características funcionales de los sistemas de información (por ejemplo, flexibilidad, reusabilidad, o independencia tecnológica). Para mostrar estas deficiencias, aplicamos herramientas de desarrollo de SIG ampliamente utilizadas y disponibles en el análisis, diseño e implementación de un SIG completo y complejo para un problema real, que está siendo utilizado en la Deputación Provincial de A Coruña (España). Analizamos también las diferencias de este SIG con respecto a la arquitectura genérica que nosotros proponemos. Dichas diferencias han sido provocadas por las limitaciones de las herramientas comerciales de desarrollo de SIG. Dicho análisis nos permite mostrar estas limitaciones que impiden el desarrollo de aplicaciones SIG con las propiedades deseables que nuestra arquitectura garantiza.

C.3 Conclusiones y Trabajo Futuro

La información geográfica se está convirtiendo en un elemento importante en los sistemas de información. Se están desarrollando muchas aplicaciones para muchas tareas industriales, administrativas y de investigación en las cuales la información geográfica es el componente central. Además, la información geográfica esta proporcionando un valor añadido a muchas aplicaciones que no la tenían en cuanta anteriormente (por ejemplo, servicios basados en la localización). Sin embargo, la información geográfica es un tipo especial de información que no puede ser representada, manipulada ni visualizada utilizando los métodos que han sido tradicionalmente utilizados para otra información de negocio y científica. La información geográfica necesita métodos de modelado y análisis especiales. La primera contribución de nuestro trabajo es un análisis exhaustivo y detallado de las características especiales de la información geográfica, y los requisitos adicionales que estas características imponen en la arquitectura de los sistemas de información geográfica con respecto a la arquitectura tradicional de sistemas de información de propósito general.

El OpenGIS Consortium y el ISO Technical Committee 211 están trabajando en un conjunto de especificaciones estándar para la representación, manipulación y visualización de información geográficas que permitirán en el futuro definir la arquitectura completa de una aplicación SIG utilizando estándares. Sin embargo, su progreso es lento y su tarea no está todavía terminada.

La contribución principal de nuestro trabajo es la propuesta de una arquitectura para sistemas de información geográfica cuyo diseño se basa en dos elementos fundamentales:

- El análisis de las características especiales que hacen los sistemas de información geográfica diferentes de los sistemas de información de propósito general.
- Las propuestas existentes del OGC y el ISO/TC 211 para modelos, servicios y lenguajes estándar para sistemas de información geográfica.

La arquitectura que proponemos es conforme con las especificaciones del OGC y de ISO donde hay estándares publicados, y proporciona detalle adicional donde no hay especificaciones. En este trabajo, hemos descrito estas propuestas, como son tenidas en cuenta por nuestra arquitectura, y las características adicionales que nuestra propuesta posibilita. Además, el diseño de la arquitectura tiene en cuenta la naturaleza y las características especiales y de la información geográfica, y al mismo tiempo, los requisitos ampliamente conocidos de los sistemas de información de propósito general.

Para validar la arquitectura propuesta, la hemos utilizado en el desarrollo de una aplicación SIG compleja para la Deputación Provincial de A Coruña. Esta es la tercera gran contribución de nuestro trabajo. Hemos presentado las aplicaciones de SIG desarrolladas para el proyecto EIEL, y hemos comparado la arquitectura de estas aplicaciones con nuestra propuesta. Las herramientas de desarrollo de SIG no satisfacen completamente los requisitos derivados de las características especiales de la información geográfica y los requisitos de los sistemas de información de propósito general, lo cual ha causado diferencias entre la arquitectura propuesta y las aplicaciones SIG desarrolladas. El uso de herramientas de desarrollo de SIG comerciales en el análisis, diseño e implementación de aplicaciones SIG causa problemas de eficiencia y falta de funcionalidad en ciertos casos. Sin embargo, nuevas herramientas como Oracle Spatial Option o ESRI Spatial Data Extender avanzan en la dirección apropiada implementando estándares internacionales para la representación de información geográfica, y proporcionando una arquitectura bien estructurada para la funcionalidad que proporcionan.

También hemos mostrado en la propuesta de nuestra arquitectura que hay mucha funcionalidad que es independiente de la aplicación particular del SIG. Esta funcionalidad puede ser implementada de forma genérica, y ser adaptada más adelante con detalles específicos descritos utilizando lenguajes de alto nivel. Sin embargo, las herramientas comerciales de desarrollo de SIG no proporcionan esta funcionalidad, y por lo tanto debe ser implementada *ad hoc* utilizando lenguajes de programación.

Los siguientes pasos sugeridos por nuestro trabajo son los siguientes:

- *Migrar el SIG de la EIEL a Oracle Spatial Option.* Oracle Spatial Option proporciona una implementación adecuada de la funcionalidad propuesta para nuestra arquitectura en el sentido de que la representación y manipulación de la información geográfica es realizada en el SGBD. Actualmente, estamos migrando el SIG de la EIEL a Oracle Spatial Option, y queremos medir la mejora en el rendimiento debido a esta migración.

- *Implementar los módulos genéricos de la arquitectura.* Aún cuando ya hemos implementado algunos de los módulos genéricos descritos para la arquitectura del SIG de la EIEL descrito en el Capítulo 5, esta implementación no es genérica porque utiliza Intergraph Geomedia Web Map. Planeamos implementar la arquitectura utilizando estándares para las interfaces del sistema, y construyendo módulos independientes de la tecnología utilizada.

- *Desarrollar herramientas para crear sistemas de información geográfica basados en la arquitectura.* Una vez que los módulos genéricos están implementados, es posible construir aplicaciones de SIG específicas conectado e integrando los módulos apropiados. También es posible y deseable definir lenguajes de alto nivel y herramientas visuales que permitan al desarrollador seleccionar e integrar los módulos de una aplicación SIG fácilmente.