Generating the Architecture of GIS Applications with Design Patterns

S. Gordillo, F. Balaguer, F. Das Neves
LIFIA-Departamento de Informática, Facultad de Ciencias Exactas, UNLP
CC 11 (1900) La Plata, Buenos Aires, Argentina
[gordillo,fede,babel17]@sol.info.unlp.edu.ar
Tel/Fax : (54) (21) 228252
*also CIC-Pcia de Buenos Aires

Abstract:

In this paper we show the impact of Design Patterns in the generation of the software architecture underlying a GIS application. We first discuss the problem of adding spatial features to legacy object oriented applications, then we present three Design Patterns specific to this domain: Reference System, Roles and Appearances to illustrate our claims.

We introduce Design Patterns as a conceptual tool both, to record design experience and to support evolvable design micro-architectures, and describe the previously mentioned patterns, exemplifying their use within the design of GIS applications. Finally, we discuss some further issues in our research.

1-Introduction

The complexity of the underlying domains of GIS applications, the variety of data types including spatial data and sophisticated relationships and the strong need of performance and accuracy in the final product, usually lead the GIS design task as a process closer to the implementation than to a software engineering process.

However, expert GIS designers do not solve every problem from scratch. Most of the time they reuse previous solutions to solve similar problems. But this reuse of experiences is difficult to transmit to non-expert designers and therefore, lacking an adequate procedure to record experience, the sharing of knowledge is not effective in helping the designer to reason in term of GIS structures; critical design decisions, such as relationships between spatial features and conceptual ones remain hidden in the code or are poorly documented. Many other decisions cannot be even deduced from either code or documentation. These strategies of reuse are caged in the designer's mind.

The problem of reusing design in GIS applications has become also a need because in the last times, more and more users are building GIS applications based on open systems instead of using a particular GIS product (like ARCInfo, GENAMAP, etc). The rapid growing of the WWW as a host for different kind of applications and the emergence of Java as an object-oriented programming language well suited for developing efficient, distributed applications, also shows us the need to find a systematic approach for reusing design experience in the context of GIS applications. This is particularly true when a designer must face hybrid applications dealing with conventional transaction-based systems and must be upgraded to include spatial features that are not built in the underlying software.

GIS 97 LasVegas Nevada USA

Examples of solutions extending traditional applications with spatial features are proposed in [8],where applications which deal with very large databases containing geographical information (terrain elevation, satellite and aerial images, detailed street maps, etc.) have to be built based on conventional applications.

Real estate agencies, telecommunications companies, hotel chains, news organizations, and survey entities usually find themselves struggling to add geographical information to their products or legacy information systems in order to enhance its ability to produce and visualize some particular data in their conceptual domain. For example, real estate agencies can browse maps finding alternatives which satisfy customer preferences (cost, distance to downtown, neighborhood style, etc.). All these applications have to support geographical queries dealing with objects within the geographical domain (downtown, neighborhood), but also with objects like houses which could have been probably defined in the conceptual domain.

Web applications providing access to geographic data captured from legacy systems are another example of this kind of applications. Developers have to write custom applications to visualize the required information. In this way we can obtain, a set of software architectures constructed in terms of design patterns that result the basis for extending object oriented applications with spatial features.

Using objects as the basis for the design of GIS applications has been recently proposed by many authors[3], [6], [11]. We claim, however, that object-oriented design methods and class libraries are only a part of the solution to the problem of GIS design. We think that the GIS community must record its design expertise in terms of Design Patterns as shown in this paper.

In the following sections we present a brief introduction to design patterns. Then we show how to use an existing design pattern Decorator [5] to incrementally construct a GIS design model. We next propose some design patterns specific to the GIS domain: Roles, Reference System and Appearance, and analyze them in the context of concrete applications. We finally motivate the need for further work in this area.

2-Design Patterns. An Introduction

Good designers use the same solution for a specific problem again and again in different applications. Recognizing a recurrent problem and using a good solution makes the difference between expert designers and inexperienced ones. Commonly, these solutions remain in the designer's mind. There are not systematic ways to record not only, the description of the problem itself but also documenting the solution, its rationale and implementation. It is clear that, this information will help another designers in the construction of other applications.

Design Patterns represent the state of the art solution for recording and reusing design experience. Basically, they describe situations that commonly occurs in an application field, and the core of the solution to these problems. Using patterns, reuse of design and architecture is not only possible but also a natural way of working.

Christopher Alexander [1] defined the purpose of Design Patterns as follows: "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice".

Design patterns are usually described by stating the problem in which the pattern may be applied, the elements that make up the design, their relationships, responsibilities and collaborations. These elements are described in an abstract way, because patterns are like templates that can be applied in many different situations. The consequences and trade-off of applying the pattern are also important because they allow evaluating design alternatives.

As it is discussed in [10] design patterns:

-Enable widespread reuse of software architectures.

-Patterns improve communication within and across software development teams.

-Patterns explicitly capture knowledge that experienced designers use implicitly.

-Patterns descriptions explicitly record engineering tradeoffs and design alternatives.

-Patterns help to transcend "environment-centric" viewpoints.

In [5] an initial catalog of design patterns is presented. We next show an approach for building the basic architecture of a GIS application relating conceptual objects with their geographic features using one of these design patterns called Decorator.

3-Using Decorators to provide Geographical features

Constructing GIS applications is not an easy task. Designers deal with two very different kinds of data types which define (in most GIS environment) at the implementation level, two separate databases: one of them contains spatial information and the other one stores the conceptual characteristics. Legacy systems that are going to be upgraded to include geographic characteristics can also be thought in terms of conceptual and geographic objects.

In some way, the complexity of the system resides in the definition and use of the spatial information more than in the behavior of conceptual objects. Our primary idea is to think in terms of the existence of two models: a conceptual and a geographical model. Objects in the conceptual model do not have the same responsibilities than objects in the geographic model, even when two objects may represent the same entity. The difference consists in the need to realize different operations in both models. In GIS applications we probably want to compute specific geographical operations (i.e. areas that were influenced by some phenomenon, entities holding a particular spatial property, etc.), but these operations are not the concern of conceptual objects like tax-payers or traffic statistic records, whose behavior is not directly related to spatial features.

Instead of modifying the conceptual classes to handle spatial features (for example by subclassing them or making them geographic classes), we can "wrap" the conceptual objects with other objects that implement the spatial handling. These objects, called "decorators", mimic the protocol of the conceptual object they wrap, but delegates the implementation of the non-spatial protocol to the conceptual object. Decorator is a design pattern defined in [5]. Its intent is to attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality. The substantial difference between using decorators and subclassing is that the former is dynamic while the later is static.The decorator conforms to the interface of the component it decorates, so that its presence is transparent to the component's clients, and augments its interface. It forwards requests to the component and may perform additional actions before or after forwarding. Transparency lets you nest decorators recursively, thereby allowing an unlimited number of added responsibilities [5].

We use the basic idea of this design pattern to construct the basic GIS model, adding spatial features to an object in a dynamic and transparent way. By using decorators, we can add functionality without modifying existing classes, at the cost of some extra coding in the decorator classes. The same scheme is useful both as a conceptual tools for new designs and to leverage existing designs to include geographic information.

Figure 1 is an example of how decorators are applied to augment conceptual classes. A Harbour class is shown here. This class was part of some existing application to track cargo ships. Lacking support for geographic information, the Harbour only knows the average depth and tide records of its surrounding bay. Later, geographic information about the harbours were available, and so it was needed to upgrade harbours to include the spatial information, but without breaking the existing class scheme. The solution is to define a GeoHarbour class, that works as a decorator of the harbour object. GeoHarbour includes the detailed harbour map contour, and adds a method able to calculate the depth of the harbour waters in different coordinates and times of the day, according the new geographic information and the tide tables.

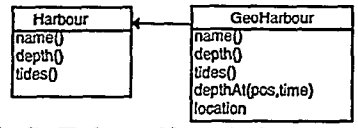

Fig. 1 : The GeoHarbour adds spatial features to a Harbour

In a more general sense, there may be objets that do not have an associated conceptual object and therefore, they only belong to a geographical model. We can think in an abstract class which groups the common behavior of those objects belonging to the GIS application model, plus those that has been wrapped from the conceptual model. Figure 2 shows this hierarchy.

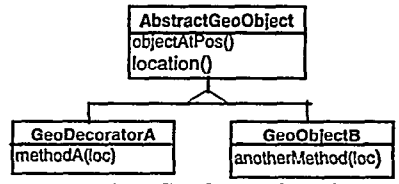

Figure 2 : A Geo-Object hierarchy.

The abstract class AbstractGeoObject defines the protocol to manipulate geo-referenced features, like behavior of general geographical functions (a point belonging to an area, perimeter, etc.). A geographical object always knows a location and, associated to the location a geometry is defined. Location has been defined as in [7] and it contains behavior about position and temporal data. Location also has a reference system and the necessary transformations to manipulate different types of geo-references. The geometry defines whether the object is represented as one or more polygons, lines, point, etc. Figure 3 shows the relationships among Conceptual classes, GeoClasses, its location, the representation and the reference system.

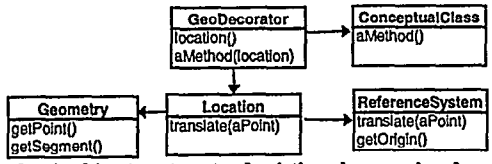

Figure 3 : Architecture to extend existing classes using decorators

This scheme is the basic building block to build the object-oriented architecture of GIS applications, composed of geo-referenced objects, some of them extending existing conceptual objects. All geoferenced objects have a location and a geometry defined as a point or a curve parameterized by a set of points. Point values are interpreted in the context of a reference system, that gives a meaning to the raw numbers with respect to an origin.

There are three ways in which this basic brick can be used as the foundation of a more comprehensive architecture for GIS applications.

- we can look deeper into the geographic objects, to think in terms of roles that defines different aspects of the responsibilities of geo-objects (Role pattern) and allows us to derive layers.
- we can explore how to build a reference systems that successfully handle coordinates (Reference System);
- finally, we can associate a geo-object with one or more visual representations. This association does not compromise the overall architecture (Appearance).

The following sections comment these patterns in detail, taking the above definition of geo-objects as a departure point.

4- Patterns for GIS design

In this section we describe three design pattern that can be used in common situation in GIS applications. We define them in terms of their intent, motivation, solution, participants, collaboration and implementation. This notation constitutes a combination of [5] and [1]. We use the OMT notation defined in [9] to describe the structure of the presented patterns

4.1-The Role design pattern

Intent:

Represent different geographic roles of an geo-object. Role lets you decouple an object by separating geographic aspects which may evolve and be used independently. Roles are the base to construct layers.

Motivation:

We are modelling a country in a geographical application. There are different aspects about a country that we may be interested about, like soils, demographic areas, climates, ozone distribution, topographic characteristics, etc. If we give to every geo-class the responsibility to maintain all the aspects it is interested about, the result is that similar code is repeated through different classes. Many times it is not easy to find a superclass to factorize the common methods, and as a result we have a set of fat classes, a poor design and future maintainability problems.

Solution:

Decoupling the responsibility of managing different aspects from the corresponding geo-object, by defining a hierarchy of roles associated to a geographical object.

Roles can be derived from other roles. Suppose that we are interested in the population of a country and its states. Then we define a role of Demographic Areas. The role Demographic Areas in a country is a derived role which takes information of the role Demographic Area assigned to states, as shown in figure 4. The country has a Derived Demographic role, that computes the total population from the population of the parts of its owner. States have an atomic demographic role, that actually keeps the number of people living in the state. Note that both derived and atomic demographic roles are not tied to Countries and States, and work with any other geo-objects that needs to keep demographic information or compute the information from other objects. Many different kinds of roles can be defined and dynamically attached to geo-objects.

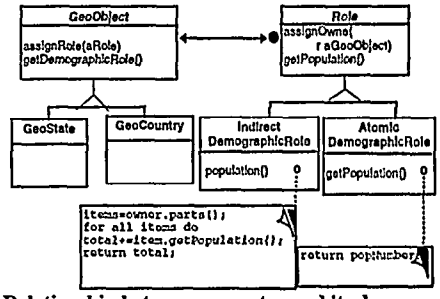

Figure 4 : Relationship between a country and its demographic role.

Structure:

The following figure shows the general structure of the role design pattern.

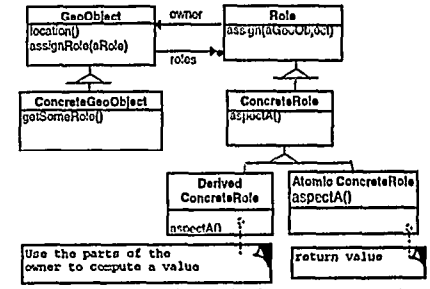

Figure 5: The Role pattern structure

Participant:

GeoObject : an object with spatial characteristics belonging to the GIS model, that may be considered playing different roles.

ConcreteGeoObject : implements a concrete spatial object.

Role: defines the abstract behavior of all roles.

ConcreteRole : a role that geographic object plays in a specific context

DerivedConcreteRole : collaborates with its owner's parts to compute a responsibility. These parts are also roles.

AtomicConcreteRole : a role that does not rely on other role objects to compute a value

Collaborations:

Geographic Object delegates the responsibility of assuming a concrete role and the manipulation of the information and behavior to the corresponding role object .

Role forwards requests to its geographic object. It may optionally perform additional operations before and after forwarding the request.

Consequences:

It decouples behavior related with a particular role from the more basic operations.

Roles allows us to define layers (see below).

4.1.1 Defining layers with roles

Roles provide the basic information to construct layers. Layers could be constructed with information from only one role or combining information from many of them and probably from the geographic object. They also could be constructed from different roles of different geographical and conceptual objects. This approach allows us to deal layers as a temporal specification, while roles define stored information in the geographical database. Moreover, we could construct layers by demand according to users' needs.

To build a layer we define a builder [5] which combines the required information of one or more roles belonging to the same or different geographic object. The builder is able to access geographic objects and roles, and generates a layer with the required information. Figure 6 shows the relationships between the builder and geographic objects.

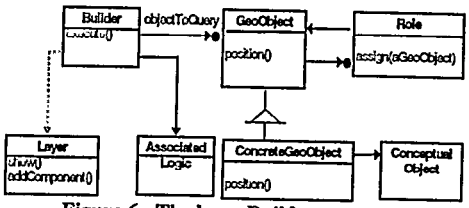

Figure 6 : The layer Builder structure

The Builder executes a number of query objects over a set of instances, in a similar way to the described in the Reports pattern [2]. The query result is a layer object that comprises a set of geo-objects, every geo-object possibly being a wrapper of a conceptual object. It is worth noting that these geo-objects are not necessarily the ones that satisfy the query. The reason is that a query can include spatial operators, like intersection or join. The result of the application of the operator to a geo-object may be a new geo-object that references the same conceptual object and reference system that the original geo-object, but that has a different geometry.

4.2-Reference System

Intent:

Decouple a measure from its reference system, which makes possible to explain the value. The reference system provides a knowledge background related to the value. Without a reference system, the measure acts like an isolated value, thus loosing testing and comparing capabilities.

Motivation:

There are different ways to represent the position of an entity on the earth. Multiple reference systems may be used to support different abstractions of the earth; these systems make possible to explain measures and operations.

For example, one of these reference systems uses elliptic coordinates; three magnitudes are required to use this system, two angular measures and one scalar. While angles represent latitude and longitude respectively; the scalar one represents the altitude from the surface of the earth. It means that a zero value represents an element on the surface of the earth; thus it is possible to model the coordinates' center like an ellipsoid.

Solution:

A solution based on Reference System is proposed. It implies making an object architecture, which contains the Location, the Reference System hierarchy, the Center and Units.

This pattern is useful for supporting multiple ways to represent geo-referenced entities of the real world. Furthermore, it makes possible to change or translate from one system to another. This change does not affect the referenced object. Each location knows the unit which is used to represent a measured value [4] For example, it is possible to build an object which has the responsibility of explaining the elliptic reference system, Figure 7 shows the referred classes and their relationships. In the same way, it is possible to model other reference systems based on spheroids or planes.

Each ReferenceSystem has to implement its legal operations such as: computing distances between points, comparing elements, calculating areas, etc.; besides it has to specify translation operations to other systems. CoordinatesCenter implements different kind of center for a specific reference system, this center can be represented such as: one point (CentralPoint) or one complex element (CentralShape)

Structure:

Figure 7 shows the structure of the Reference System pattern.

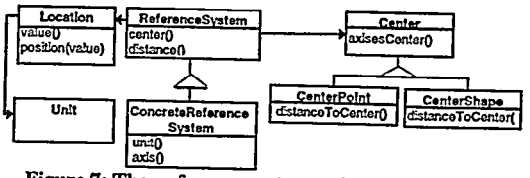

Figure 7: The reference system and its relationships

Participants:

Location: implements the basic behavior to support positioning. It can explain where an entity of the real world is.

ReferenceSystem : describes an abstract protocol, which is used to describe the context where a Location is defined.

CoordinatesCenter: models the center of the reference system. The coordinates center is a characteristic aspect of each kind of *ReferenceSystem* : generally it would be a point or a complex shape, which is defined by an equation.

Center : is the object which represents the coordinate center of the reference system

Unit : it is used to represent a measured value

Collaborations:

Reference System provides a set of valid operations. Each time Location needs to use geo operations, it asks for these operations to Reference System. Location provides an unified protocol to represent location within a specific reference system.

Consequences:

-It decouples measures (such as positions) from the reference system being used.

-If only one reference system is used or if it is 1-dimensional, the pattern produces an overhead because it works within fine granularity abstraction.

- Instances of the ReferenceSystem hierarchy can be implemented following the Singleton pattern.[5]

4.3-Appearance

Intent :

Produce visual representations of geo-objects independently of their geometrical representation.

Motivation:

Usually, geographic information systems produce a graphical output of the information. The way in which geographical objects are perceived in the user interface depends on the user's preferences. Moreover, the representation chosen by the user could be different from the geometrical representation defined in the system.

Looking at the relationships between conceptual objets and GeoObjetcs expressed in section 3, a first attempt to solve this problem would consider assigning the representation responsibilities to the GeoObject thus considering it as a conceptual object's view. The problem is that the same conceptual object may have many different views in different user interfaces.

For example, while the geometrical representation for a ship .in a delivery application is a point, we define an user interface where the ship is seen as a silhouette and another one where it is seen as a point moving on the water.

Solution:

Consider the representation as a separated object , and associate it to the GeoObject by a Model/View [5] scheme. This object knows how to display the GeoObject, according to some appearance properties, such as color, definition level, line thickness, etc.

Structure:

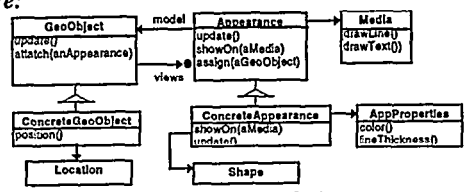

Figure 8 : the Appearance design patterns

Participants:

AbstractGeoObject: Defines the common protocol for all geo-referenced objects (see section 3).

ConcreteGeoObject: Extends the protocol of GeoObject for specific geo-referenced entities

Appearance: Speficies the protocol to receive notification of changes from the model object, and to display an object in a particular media

AppProperties: Abstract class that defines the set of properties that modifies how the Shape is displayed by the Appearance object

ConcreteAppearance: Implements and redefines the protocol of Appearance to display ConcreteGeoObjects through a Media. Objects according to a set of properties.

Media: Implements the operations needed by ConcreteAppearances to display GeoObjects in different media (text stream, graphic devices, etc.)

Shape: An Arbitrary object whose protocol is known by the ConcreteAppearance object, and that the later is able to display

Collaborations :

All GeoObjects that are going to be displayed have one or more associated Appearance objects, that are recorded by the GeoObjects by sending the attach() message. An Appearance object knows the GeoObject that works as its model when it receives the assign() message.

When Appearance objects are created, they get a Media object assigned. The Media object knows how to draw or write in a specific media, like a bitmap or text stream.

ConcreteGeoObjects notify changes in their state to their Apearances. The ConcreteAppearance object calls one or more methods in the Media object to create the representation.

Consequences

By decoupling the representation from the geometry, the Apearance Design Pattern simplifies the specification of multiple views of the GeoObject.

5-Discussion and further work

We have presented some design patterns generating the architecture of GIS applications. Systematically applying the solutions discussed above we can build evolvable applications that combine both spatial features with more conventional object-oriented behavior. Using "GeoWrappers" results in an architecture where non-spatial features are decoupled from geographic ones thus allowing them to evolve independently. Reference System provides a knowledge background related to spatial values, Role allows us to define contexts in which different information results of relevance, finally, Appearance is used when we want to separarte representation from geometry.

We have implemented them using the Gemstone/VisualWorks object oriented database management system and found them rather useful both as the basis for extending legacy o-o applications with geographical features and to build new, modular applications from reusable components. We are now extending our approach by defining new general patterns, like Isopleth which allows us to group the information and behavior to construct curves connecting points with a specific logic. We are also studying new patterns that appear while using the WWW as the infrastructure for GIS applications.

6-References

[1] C. Alexander, S.Ishikawa, M.Silverstein, M.Jacobson, I.Fiksdahl-King and S.Angel : "A Pattern Language". Oxford University Press, New York, 1977.

[2] J. Brant and J. Yoder : "Reports", Proceedings of PLoP96, Conference of Pattern Langages of Programming. Washington University Technical Report WUCS-97-07

[3] Pei Min Chen, Shou Yi Tseng, Young Chang Hou and Bin Bin Loah: *"An Object-Oriented Geographic Information System Shell"*. Proceedings of GIS'95, Vancouver, Canada, pp 413-420.

[4] M.Fouler : "Patterns : Reusable Object Model". Addison Wesley, 1997

[5] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *"Design Patterns. Elements of reusable Object-Oriented Software"*. Addison Wesley, 1995.

[6] C. B. Medeiros, M. A. Casanova and G. Camara: *"The Domus project. Building an OODB GIS for environmental control"* Proceedings of IGIS'94. International Workshop on Advanced Research in GIS, Springer Verlag LNCS, N. 884, pp 45-54

[7] Open GIS Consortium (OGC) (1996b), The Open GIS Guide - A Guide to Interoperable Geoprocessing, Available at http://ogis.org/guide/guide1.html

[8] M.Postmesil : "Maps Alive : Viewing Geospatial Information on the WWW". Proceedings of the Six International World Wide Web Conference, 1997. Available at http:// www6.nttlabs.com/Hypernews/get/PAPER130.htm

[9] J. Rumbaugh, M. Blaha, M. Premerlani and W. Lorensen: *"Object-Oriented Modelling and Design"*. Prentice Hall, Englewoods Cliff, New Jersey, 1991

[10] D. Schmidt: *"Using Design Patterns to Develop Reusable Object-Oriented Communication Software"*. Comm. of the ACM, October 1995, pp 65-74.

[11] N. Tryfona and T. Hadzilacos: *"Geographic Applications Development: Models and Tools for the Conceptual Level"*. Proceedings of ACM-GIS'95. Baltimore, Maryland, USA, pp 19-28.