Designing Adaptable Geographic Objects for Mobile Applications

Robert Laurini
INSA-Lyon, France
Robert.Laurini@lisi.insa-lyon.fr

Javier Bazzocco
LIFIA, Facultad de Informática,Universidad Nacional de La Plata, Argentina
jb@lifia.info.unlp.edu.ar

Silvia Gordillo
LIFIA, Facultad de Informática,Universidad Nacional de La Plata, Argentina
gordillo@lifia.info.unlp.edu.ar

Gustavo Rossi
LIFIA, Facultad de Informática,Universidad Nacional de La Plata,Argentina
gustavo@lifia.info.unlp.edu.ar

Catalina Mostaccio
LIFIA, Facultad de Informática,Universidad Nacional de La Plata, Argentina
catty@lifia.info.unlp.edu.ar

Abstract

In this position paper we discuss the evolution of an object-oriented framework for GIS applications into a platform for dealing with mobile users and geographic objects. We first motivate our research by discussing the state of the art of mobile GIS applications, the so-called location-based services using a simple scenario. We briefly describe an object-oriented architecture for dealing with discrete and continuous geographic data and show how good design practices allowed us to make geographic objects adaptive to mobile users. Some concluding remarks are finally presented.

1. Introduction

In the last 5 years, we have experienced an increasingly interest in the development of ubiquitous applications, i.e. those applications that follow the anytime/anywhere/any media paradigm and provide transparent access to information and other kind of services trough different (in general portable) devices. Mobile applications are one important type of ubiquitous software; these applications have the ability to adapt themselves to the user's context, e.g. his location, the device he is using (a laptop, palm computer, cell phone, etc), his preferences, etc. Research issues related with mobile computing range from hardware (small memory devices, interface appliances) and communication networks (trustable connections, security, etc) to software and data management aspects such as new interface metaphors, data models for mobile applications, continuous queries, adaptive applications, information exchange between disparate applications, etc. In this position paper we deal with a particular kind of mobile applications, those that adapt their services to the user's location, the so-called Location-Based Services.

Location-Based Services evolve in a similar way as the more generic class of ubiquitous applications. According to Abowd [1]: "Ubicomp applications evolve organically. Even though they begin with a motivating application, it is often not clear up front the best way for the application to serve its intended user community". As a consequence design issues are critical for the application to evolve seamlessly when requirements change [2]. In our research, we are pursuing the definition of a modular design approach for building location-based applications. In particular, we have identified a set of design micro-architectures to build evolvable location models, i.e. those application components that represent the user location and which are used to adapt the application's behavior accordingly [3]. However, location-based services might also involve the interaction with legacy GIS (geographic information systems) software. This interaction poses new architectural and development challenges to the

Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops (WISEW'03)
0-7695-2103-7/04 $20.00 © 2004 **IEEE**

designer; for example if we are using a commercial (monolithic) GIS product, we may need to wrap its functionality to use it in the mobile context; even with open architectures such as those defined by [20] we must write some adaptation code to deal with the location's context. In this position paper, we discuss some lessons learned while re-designing an object-oriented framework for GIS software into a platform for dealing with location-based services. Our contribution is twofold: first we reflect on our experience to indicate which design practices help in the process of evolving software systems into position-aware systems; second we identify a set of problems (and corresponding solutions) for dealing with geographic objects in the context of mobility.

The structure of this position paper is as follows: In Section 2 we survey the state of the art of Location-based software, analyzing their evolution from monolithic GIS applications to lighter Internet services and introduce an example scenario. In Section 3 we describe our GIS framework and analyze which design challenges we had to face to make it user-aware. In Section 4 we discuss the core of our solution and discuss how to integrate the evolved architecture with adaptable (object) location models. Finally, in Section 5 we present some further work and concluding remarks.

2. From GIS applications to Location-Based Software

Geographic Information Systems were originally thought to deal with spatial data to provide information related to situations of terrestrial objects and predictive analysis to study phenomena evolution. GIS software usually provides powerful functions and interfaces in order to calculate and provide the result of spatial analysis [17], [19]. GIS technology has evolved from mainframe-based applications to desktop and, recently, distributed information systems. A nice description of this evolution can be found in [21]. This evolution has been accompanied by a corresponding evolution in the software modeling and design techniques related with the construction of these (complex) applications. The era in which GIS applications were closed, proprietary repositories of tangled geographic data queried with ad-hoc techniques has ended. We can now use light geographic services provided by public APIs, and open source GIS applications that support a wide variety of spatial data functions. The advent of the Internet, the use of the WWW as a platform to deliver multimedia information and to use sophisticated information systems, has given a new opportunity for disseminating and popularizing the use of GIS. Web Cartography [15] and cartography services [24] are now widely used. New architectural approaches to build service-based applications (using Web services for example) gave more impulse to this trend and the emergence of standards for interoperability such as GML [6], [7], made it feasible.

Location Based Services (LBS) can be seen as the logical evolution of geographic applications in the context of mobility [12]. Mobile GIS applications allow the use of geographical data from wireless devices such as palm and pocket PCs (eventually with accompanying positioning devices like GPSs). In [21] LBS are defined as "applications that have geospatial data-handling functions and the integration of geo-referenced information with other types of data. For example, car navigation systems, realtor systems and pizza delivery are some representative location-based services. Mobile GIS has become the perfect platform for the development of comprehensive location-based services". However, and even taken into account that the root of LBS is GIS software (and that from the functionality point of view, a GIS module is necessary to build a LBS), the evolution patterns of LBS are clearly different from those of GIS software, and accordingly the software design requirements change dramatically as discussed in [3]. Different architectures and design solutions have been presented in [12] and [13] to deal with the mobility issue; nevertheless, the subject of integrating GIS software with ubiquitous computing from the object oriented point of view has been recently introduced and studied in [3], [8] and [10]. In this paper we address a more complex problem when dealing with GIS software in the context of mobility: how to adapt the structure, representation, topology and behavior of geographic objects when the user moves.

To make this discussion concrete, suppose for example a simple application to provide the user with tourist information while he moves throughout a country like France using his preferred device. While in the highway, he is prompted with information about best routes to go somewhere, he is informed about tourist spots and services (like gas stations), etc. When he enters a city like Paris he can be told how to go to a place from where he is now, which hotels and restaurants he can find in the neighborhood, etc. Existing state-of-the-art technologies such as positioning devices and Internet cartography [15] make this scenario absolutely feasible. When he enters a Museum the problem has a new shift. Using a new set of positioning artifacts like beacons [22], we can eventually know in which room he is, and we can tell him how to go where he wants. If we are able to know the artwork the user is watching (another kind of "location"), we may want to explain him some facts about its author, the historical context, etc. While most technological requirements in this scenario can be easily fulfilled, there are many design and usability problems that need some further study. We will focus on one of these design problems; how to adapt the basic features of geographic

objects to the user location and context, e.g.: while he is in the highway, Paris can be represented by a point geometry [17]; in the latter examples we need a polygon geometry in which Museums are points; finally we need Museum to be represented as a polygon. In the rest of this paper we refine this discussion.

3. A Framework for GIS Applications

In this section we briefly introduce our object-oriented framework for building geographical applications (Geo-Framework). This framework is fully described in [8], [9], [10], and [18]. Geo-Framework provides a set of basic classes and abstract behaviors that can be either extended or customized for specific applications (and domains).

The main architectural design decision of the framework is a clear separation between application objects and their spatial features, described as shown later as decorations [5]. This approach makes possible not only the development of new applications decoupling complex concerns like the definition of spatial features, but also the extension of "conventional" information systems with geographic features, a problem that is outside the scope of this paper (even though some of the solutions proposed here share the same design philosophy).

3.1. The Base Architecture

When designers build GIS applications, they deal with two different kinds of data types. One represents conceptual data in terms of descriptive attributes. For example, when modeling a country in the context of a geographic application, typical descriptive attributes are the name of the country, its first language, or its government system. The other kind of data represents all aspects related to geographic features, like the country boundaries, its position in the map, etc. Designers have to deal with different aspects of the same application object: the specification of conventional data (like name of the country) and behavior (operations like tax-payment), and the definition of spatial information and spatial analysis.

Geo-framework induces a structure in which an object-oriented model of the application is built first without considering spatial features. Geographical aspects and behaviors are added by first identifying which classes in the application model will contain spatial features; then, for each one of these classes, we define a new one that wraps it with the spatial behavior by applying the Decorator design pattern [5]. As an example, we suppose that we have a Country class and decorate it to include the spatial information, in order to perform operations like width in a particular latitude, its location, neighbor countries and so on. Figure 1 shows both conceptual and geographic definitions; additional features have been defined in the second one (abstract classes are not shown for simplicity).

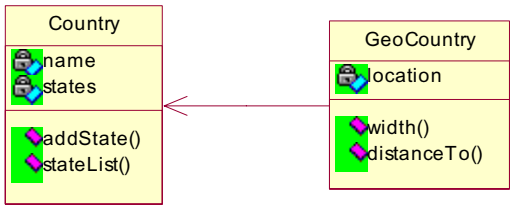

Figure 1. Conceptual and geographic classes

Geographic objects, have a location attribute which abstracts positioning information; in particular its topology (point, line, polygon) and its reference system. In this way, each spatial object will have associated a Location class which, in turn, is related to a Topology and ReferenceSystem classes shown in Figure 2.

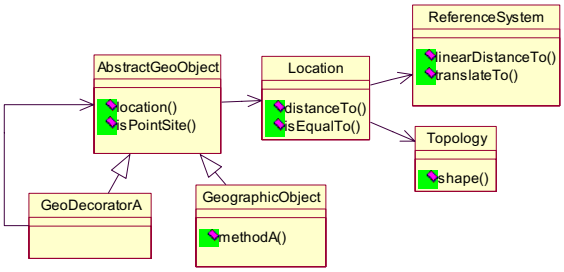

Figure 2. Geographical Classes, their topologies and reference systems

Topology is a class hierarchy defining the basic spatial elements: point, line and polygon. In this hierarchy, all spatial operations are defined according to the kind of elements we are manipulating. For example: one element defined as a polygon will be able to perform operations such as adjacency, intersections, inclusion, etc. ReferenceSystem meanwhile defines an abstract protocol that is used to describe the context where a Location is defined. It also defines the set of legal operations in this context. In other words each ReferenceSystem instance describes how measures are interpreted in the defined Location.

The framework also supports dealing with "pure" geographic objects such as Continuous phenomena or fields (the temperature in a country, the level of pollution of a river). Each field is usually described in terms of a set of elements such as: the nature of the field, dimensions, an interval of dates and a set of points representing the sample which makes the field discrete and computable. Each point belonging to the sample, in turn, contains its position (also represented by the Location class) and a value. Samples may have different implementations which is achieved by using an instance

of the Bridge pattern [5]. Finally, estimation methods for calculating the value of the field in a particular position are provided using Strategies [5]. In this way both representations and estimation methods can be changed dynamically during run-time. Figure 3 shows the sub-architecture for dealing with continuous fields.

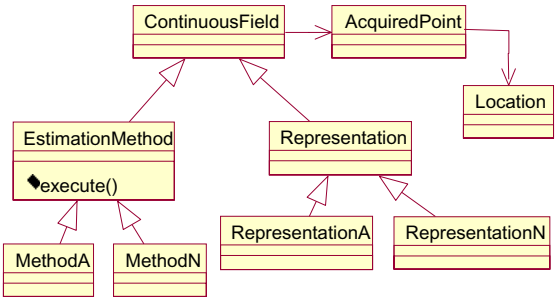

Figure 3. Sub-architecture for continuous fields

3.2. Design challenges with mobile users

We have used this framework to instantiate applications in different fields such as Agriculture Intelligent Systems and AVL (Automatic Vehicular Location). In the former case we dealt with moving geographical objects whose current position should be found or estimated. Decoupling conceptual from geographical objects, and these objects from their positions allowed us to easily add behaviors to track (or simulate) movement. However, when the user himself is moving some new and challenging problems might appear. For the sake of conciseness we assume from now on that users are mobile and geographic objects static.

The first problem, as shown in the example in Section 2 is related with the topology of geographic objects while users move. Typical GIS applications and even Location-based Services define only one topology (and thus, reference system) for a geographical object. This means that if Paris is represented as a point, this representation can not change to a polygon dynamically. A naïve solution would be to use always the most generic one (a polygon) and perform the adaptation only at the interface level. However, as geometric operations and constraints are based on the chosen topology we might have problems. Moreover, we might need another topology for objects with a local reference system, such as a museum and as the kind of queries users pose to these geographical objects change with his location (e.g.: how do I reach the museum? , where is artwork x?) we might end with classes supporting a huge number of operations. Summarizing: in terms of design problems, we need that the relationship among Geo-Objects and their locations (in Figure 2) are mediated by the user position (or more generally by the user context).

It is easy to see that the same problem appears with other relationships such as location and reference system (for example a pair of numeric attributes might have different semantics in different referent systems). We do not discuss here variations related with continuous fields to keep the examples simple.

4. Towards Mobile and Ubiquitous GIS

Applications built from Geo-Framework can be used from Web-compliant interfaces by using the Geographic Mark-up Language (GML) for sending geographical objects/interfaces as responses to http requests [11]. We also built an infrastructure for using some specific Geo-Framework functionality using Web Services [16]. In the next sub-sections we describe the key architectural decisions in order to support mobile users.

.

4.1. The Architecture of a mobile GIS application

Generally speaking, the kind of applications we want to build usually need an adaptation capability (that "conventional" GISs do not have) to modify their behaviors according to the user position. Notice that we use the term "behavior" even to indicate the special case in which we need to change a relationship.

Existing arquitectural solutions for the design of ubiquitous (Web) software, can be applied in this field with minor modifications. For example we can use the approach described in [13] in which three important architectural components are described (See Figure 4):

-the application model containing main application classes and functionality; it must be constructed to be independent with respect to types of users and adaptation rules.

-the user or context profile: that contains information about the users' interests and preferences and the actual usage context; in particular, this module is responsible of maintaining the current user's location.

-the adaptation model that encapsulates different kinds of rules, for example for adapting the application behavior to specific contexts or situations.

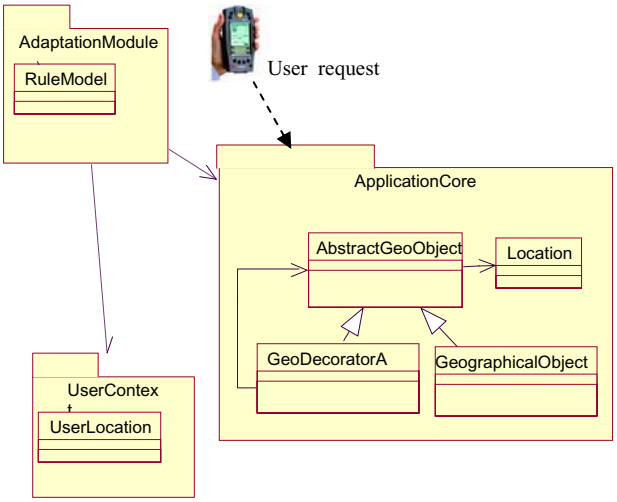

Figure 4. The generic architecture of Ubiquitous Applications

When using this architectural strategy, domain model behaviors that need to be adapted are mediated by the rule (adaptation) model that, collaborating with the corresponding user profile objects, decides how the behavior is affected. A clear separation between the adaptation model and the user profile, decoupling them from the application model allows easier maintenance and prevents the core functionality from being cluttered with conditional sentences regarding user's conditions and usage context. The cost of this solution is that the rule model might become too complex with hundreds of rules that must be maintained and kept up to date. In [4], we have discussed when adaptation rules should be replaced by polymorphic behaviors in order to simplify the adaptation model. In the following sub-sections we show how we adapted Geo-Framework to this scheme.

4.2. Adaptable geographic objects: Wrappers to the rescue

The first critical decision is how to easily trigger the adaptation model when a user request arrives. In [23] we show how to use wrappers to seamless customize existing applications. We chose a similar approach: for each behavior (or set of behaviors) that should be adapted we built a light weighted object that provides the modified behavior, in this case by sending a message to the corresponding adaptation rule, as shown in Figure 5. This approach helps us to adapt behaviors in an instance-based and dynamic way, without paying the cost of massive class modifications.

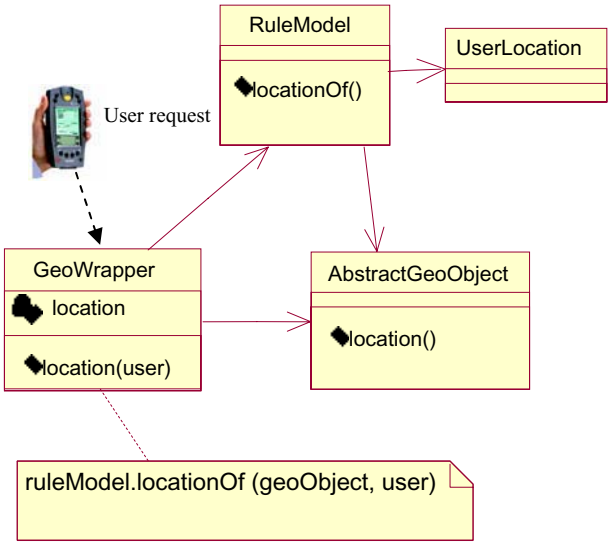

Figure 5. Triggering Geo-adaptation behaviors

Notice that, from now on, the topology of a geographic object is not defined by a static relationship but instead it is represented by a rule that connects the location object with its corresponding topology and reference system. In some cases it might be necessary to wrap more than a simple behavior in the chain leading from the request to the object that provides the specific answer.

This solution, while fully compatible with the architecture of Figure 4 has a problem (that we discussed in [4]); it over-emphasizes rules for providing the corresponding behaviors. A typical adaptation model implementation will provide condition and action objects, the former ones for querying the user model and the latter ones for performing the corresponding action.

We found that a slightly different solution is better for performing adaptations related with user locations: moving the adaptation behavior to the user model instead of viewing this model as just a repository of data about the user. With this strategy, different location contexts might provide different (adaptation) behaviors. This approach is shown in Figure 6.

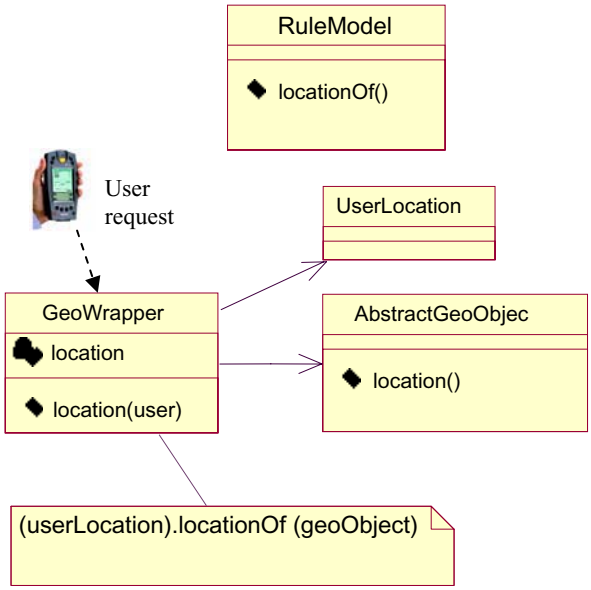

Figure 6. Allocating the adaptation code in the user model

This solution has a problem: objects that represent user locations might have different attributes according to the positional system we use. It may be not possible or reasonable to define new classes each time the application evolves. We solve this problem by using an extension of the type object pattern (known as Adaptive Object Models) [25].

4.3. Representing mobile users with adaptive object models

If we consider that different user location contexts can be expressed in a different reference system (e.g. a position in the highway using latitude/longitude, a location in the Museum as a room number), it is quite clear that a solution based on inheritance and sub-classing is not the best one because me might end with a large number of different classes with minor structural differences but with few or none behavioral refinements between them. We thus replace different user location classes with a generic class LocationType whose instances are different types of locations as shown in Figure 7. Each Location type defines a set of property types, having a name and a type (class PropertyType). Instances of Location contain a set of properties (instances of class Property) each one referring to one property type. Using the "square" in Figure 7 we can manage the meta (or knowledge) level by creating new instances of the "type side" (at the right) and the concrete level by creating new instances of classes in the left.

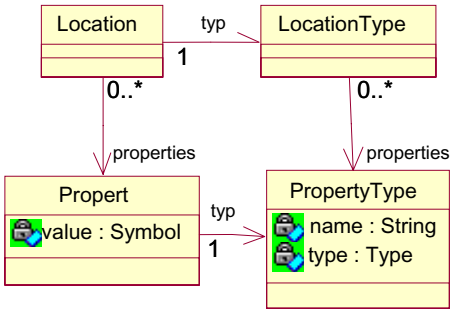

Figure 7. Adaptive model for locations and their properties

By this mean, adding new types of locations is not restricted by the code, compile & deploy process (which is still error prone in languages such as Java), which is known to be a very "static" solution. By using the preceding approach, the definition of a new kind of location can be easily made by arranging the required properties instances as needed (each one of them belonging to a particular type of property). The static definition of the structure imposed by the classes approach is changed in favor of the more dynamic alternative presented by the "square" solution presented above. From the "code-level" point of view, the presented approach has some important benefits: there is no need to create a large set of location sub-classes that only differ in their structure; additionally, the design leads a greater utilization of the polimorphism since no distinction is made regarding the type of the location (every location object is an instance of the same class, but configured in a different way).

Regarding adaptation we need that different Location objects (and of course Location types) might provide different geographic adaptation behaviors. For expressing simple adaptations (for example a change in the topology or reference system) we simply connect the location object with the corresponding geographical object. For more complex adaptations (being them geographic or not) we used Strategies [5] that are attached to the corresponding Location object as shown in Figure 8. Geographic strategies can be implemented using conventional algorithmic style or they may be described as geographic rules using the infrastructure of the rule model. The decision depends on the kind of adaptation that should be performed; for example, if the behavior might be cluttered with *if* sentences, the rule style is preferable.

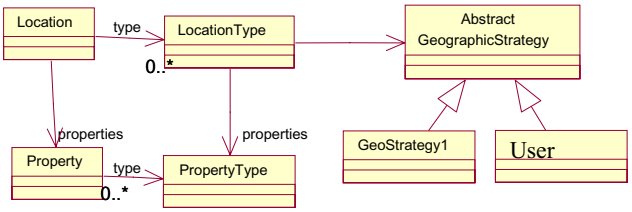

Figure 8: User Location objects and their behaviors

5. Concluding remarks and further work

In this paper we have presented some micro-architectures that arose when re-designing a large object-oriented framework for GIS applications to make it sensitive to mobile users' requests. Our experience have shown us that the evolution from "conventional" GIS software into location-based services presents many design challenges that are seldom discussed in the literature, mainly because technological changes are still occurring and are receiving much more attention.

We have identified a set of specific problems related with the geometry, reference system and other geographic features, when the user moves. Having different "views" of a geographical object is often over-simplified to user interface problems (seeing a city as a point, polygon, etc); however, we have shown that adapting geographic objects to the needs of a mobile user requires quite more subtle changes. In this paper, we have discussed some of these problems and we have shown that seamless extensions to the Geo-Framework were easily achieved by using good object-oriented design practices; we strongly believe that these practices are in the heart of every evolvable mobile software. We also presented an original approach for dealing with geographic adaptations by introducing adaptive object models as the key solution for representing the user's location.

We are now researching on some issues related with the combination of geographic with more typical information systems behaviors in the context of mobility. In our example in Section 2 we want that the user is provided with different functionality as he moves. In our modified framework this is possible by adding those behaviors to location objects; however higher level objects (with more semantics than just the user's position) are necessary: we call them location contexts. Location contexts are the user profile counterpart of some application objects (a city, a museum, etc); they are described as the complement of the corresponding local Reference System that (as explained in Section 3.1) contains the basic geographical operations. We are exploring the type of architectural connections we must build between both kinds of objects. At the same time we are adapting some of our micro-architectures to well-known styles such as the Model-View-Controller [14] to simplify the connection with existing middleware, that in fact is responsible of identifying the request, the current user, his location and location context.

6. References

[1] G. Abowd, "Software Engineering Issues for Ubiquitous Computing", *Proceedings of the International Conference on Software Engineering (ICSE 99)*, ACM Press, 1999, pp. 75-84.

[2] G. Banavar, and A. Bernstein, "Software Infrastructure and design challenge for ubiquitous applications". *Communications of the ACM*, Vol 45, (12) pp. 92-96.

[3] J. Bazzocco, S. Gordillo, G. Rossi, and R. Laurini, "Designing Evolvable Location Models for Ubiquitous Applications". *Proceedings of OOIS*, Lectures Notes in Computer Science, Springer-Verlag Heidelberg, ISSN: 0302-9743 , 2003, Volume 2817, pp. 289-293

[4] J. Cappi, G. Rossi, and A. Fortier, "Customization policies need more than rule objects", *Proceedings of OOIS 2002*, Springer Verlag, Lectures Notes in Computer Science.

[5] E. Gamma, R. Helm, J. Johnson, and J. Vlissides, *Design Patterns. Elements of reusable object-oriented software*, Addison Wesley, 1995.

[6] A. Garmash, "A Geographic XML-based Format for the Mobile Environment", *Proceedings of the 34 th. Hawaii International Conference on System Sciences*, IEEE Press, 2001.

[7] Geographical Mark-up Language, in www.opengis.org/techno/specs/00-029/GML.html

[8] S. Gordillo, F. Balaguer, and F. Das Neves, "Generating the Architecture of GIS Applications with Design Patterns", *Proceedings of the ACM-GIS97: 5th International Workshop on Advances in Geographic Information Systems*, Ed. R. Laurini, P. Bergougnoux, K. Makki and N. Pissinou, 1997, pp. 30-34.

[9] S. Gordillo, and F. Balaguer, "Refining an object-oriented GIS design model: Topologies and Field Data", *6th ACM Workshop on Geographic Information Systems*, Maryland, USA, 1998, pp. 76-81.

[10] S. Gordillo, F. Balaguer, C. Mostaccio, and F. Das Neves, "Developing GIS Applications with Objects: A Design Pattern Approach". *GeoInformatica*. Kluwer Academic Publishers, 1999, Vol 3:1, pp. 7-32.

[11] S. Gordillo, *Modélisation et Manipulation de Phénomènes Continus Spatio-temporels*, PhD, Université Claude Bernard, Lyon I, 2001. 190 pp.

[12] J. Hjelm, *Creating Location Services for the Wireless Web: Professional Developer's Guide*, John Wiley, 2002.

[13] G. Kappel, B. Proll, and W. Retschitzegger, "Customization of Ubiquitous Web Applications. A comparison of approaches", *International Journal of Web Engineering and Technology*, Inderscience Publishers, January 2003

[14] A. Knight, and N. Dai, "Objects and the Web", *IEEE Software* ,March-April 2002, pp. 51-59.

[15] M. Kraak, and A. Brown (editors), *Web Cartography. Development and Prospects*, Taylor and Francis, 2001.

[16] F. Krimerman, and A.Glavina, *Designing Spatial Web Services*, (in spanish) Universidad de Buenos Aires, Argentina, 2003.

[17] R. Laurini, and D. Thompson, *Fundamental of Spatial Information Systems*, Academic Press, 1993.

[18] R. Laurini, and S. Gordillo, "Field Orientation for Continuous Spatio-temporal Phenomena". *International Workshop on Emerging Technologies for Geo-Based Applications*, Ascona, Switzerland, 2000, pp. 77-101.

[19] P. Longley, M. Goodchild, D. Maguire, and D. Rhind, *Geographical Information Systems and Science*, Wiley, 2002.

[20] Open GIS Consortium, Inc. 2003 in www.opengis.org

[21] Z. Peng, and M., Tsou, *Internet GIS. Distributed Geographic Information Services for the Internet and Wireless Networks*, John Wiley, 2003

[22] S. Pradham, "Semantic Location. Personal and Ubiquitous Computing". Springer Verlag 2002, Vol 6, pp. 213-216.

[23] G. Rossi, A. Fortier, J. Cappi, and D. Schwabe, "Seamless Personalization of e-commerce applications", *International Workshop on E-commerce and conceptual Modeling*, Springer Verlag, Lectures Notes in Computer Science, 2001,

[24] K. Virrantaus, J. Veijalainen, and J. Markkula, "Developing GIS-Supported Location-Based Services", *Proceedings of the Second International Conference on Web Information Systems Engineering (WISE'02)*.

[25] J. Yoder, and R., Razavi, "Metadata and Adaptive Object-Models", *ECOOP 2000 Workshops*, in www.adaptiveobjectmodel.com.