Clodoveu Augusto Davis Jr.
Karla Albuquerque de Vasconcelos Borges
PRODABEL -- Processamento de Dados do Município de Belo Horizonte
Av. Presidente Carlos Luz, 1275
31230-000 -- Belo Horizonte -- MG
BRAZIL

# OBJECT-ORIENTED GIS IN PRACTICE

**Abstract:** Much has been said about object orientation, and how it should change the face of information technology. But what would be the real benefits of using an object-oriented GIS? This paper presents a set of impressions and opinions gathered in the last two and a half years, in which the authors have been using a GIS based on an object-oriented database, to manage 5,000,000+ objects and their attributes. Basic concepts in object technology, such as inheritance, polymorphism and encapsulation are reviewed, and their meaning is transported to the needs of a GIS. Design issues are also addressed, covering conceptual, logical and physical aspects. General user comments, in areas such as reliability, flexibility and ease of use are issued, as well as a "wish list" of desired features that should appear in future OOGIS (object-oriented GIS) implementations.

## INTRODUCTION

The abstraction of concepts and entities that can be found in the real world is an important part of the creation of information systems. Moreover, the success of a computer implementation of an information system depends on how well the real world entities and their interactions are transposed to a computer database. Abstraction functions as a tool to help us comprehend the system, by dividing it in separate components, that are viewed in different levels of complexity and detail, according to the need to understand and represent the various real world entities and their interactions (Hughes, 1991). The need for better abstraction tools in database management systems (DBMS) is one of the determinant factors of the development of object-oriented technology.

There is a great distance between sophisticated information structures, as demanded by some types of applications, and the simplicity of the data model in which conventional database systems allow these structures to be represented (Worboys *et al.*, 1990). This generates all kinds of difficulties, including loss of performance and excessive disk space consumption. Non-conventional database applications, such as multimedia, computer-aided design (CAD) and geographic information systems (GIS) are beginning to take advantage of the stronger logical structure provided by object-oriented

database management systems (OODBMS) (Stone and Hentchel, 1990). These applications have the need to manage potentially large volumes of non-conventional data, like images or geographic coordinates, using data structures that carry a better resemblance to the user's universe.

The importance of conventional DBMSs in today's world, however, cannot be diminished, since they are used to manage most of the available data in every organization, and will remain useful for many years. It is hardly questionable, though, the inadequacy of using conventional DBMSs to manage spatial information.

## BASIC CONCEPTS

An *object* is a structure that represents a single entity, describing both its information contents and its behavior. Every object belongs to a *class*, which defines a structure and a set of operations that are common to a group of objects. Individual objects of a given class are often referred to as *instances* of the class.

An object functions as a complex data structure, that is capable of storing all of its data, along with information about the necessary procedures to create, destroy and manipulate itself. In an object-oriented GIS (OOGIS), each object instance would contain its graphical characteristics, its geographic location, and all of the associated data.

The ability to hide from the user the internal structure of an object is called *encapsulation*. With encapsulation, it is only possible to manipulate the object’s data using a set of predefined functions, thus ensuring data independence. The internal definition of the data structure can then change, without influencing what the user perceives (Edelstein, 1991).

Object classes are often defined hierarchically, taking advantage of one of the most important concepts in object-oriented systems: *inheritance*. It is possible to define more general classes, containing the structure of a generic type of object, and then specialize this class by creating subclasses. The subclasses will inherit all properties of the parent class and add some more of its own. For instance, when creating an object class to represent water network nodes, one might define subclasses to be `pump` and `reservoir`, which would inherit generic water node characteristics such as `code` and `location`, along with their relationship with water pipes, and aggregate specific characteristics such as `pump throughput` and `reservoir capacity`.

Another important feature of object orientation is called *polymorphism*. This refers to a program's ability to use objects belonging to different classes in a transparent way, interpreting their characteristics on the run. For instance, if a program needs to have access to an attribute called `code`, that has been defined for several different object classes, or for classes with common ancestry, it is not necessary to know the object's class beforehand.

## DATABASE MODELING WITH OOGIS

Relational DBMSs generally work very well with tabular data, in which the structure is very simple, or is made simple by some design technique. This is not the case of spatial data. When using a relational DBMS to manage spatial data, users are often forced to transform the idealized data model to fit the relational model's requirements, creating artificial restrictions such as normalization. Because of these transformations, the performance of the system is degraded, becoming unacceptable when used with large amounts of data (Egenhofer and Frank, 1992).

GIS users, moreover, often need to integrate data from different sources, using the lowest common denominator usually available: geographic location. Therefore, flexible data models are demanded, in order to accommodate the heterogeneity, while maintaining the possibility to perform several usual and unusual tasks. A GIS should be able to represent graphically the real world with the sophistication of a CAD system, while allowing users to analyze the available data in different conceptual levels, with varying degrees of complexity.

The creation of a data model for OOGIS is still mostly uncharted territory. Usual data modeling techniques such as entity-relationship quickly show their limitations when exposed to the needs of spatial data management. Extensions to the entity-relationship approach have been proposed, and there are also a number of techniques based on semantic data models. Only further research, combined with more experience, will demonstrate which is the best approach (Gray *et al.*, 1992).

## CONVENTIONAL VERSUS OBJECT-ORIENTED GIS

Generally, GIS software use a combination of graphics files and a relational database. Some go even further, storing both graphics and data in the relational DMBS. Considering what has been said about conventional databases and GIS, using an object-oriented approach should have, theoretically, at least these advantages:

- **Data integrity:** since both graphics and data are maintained in the same data structure, the integrity is ensured (Wang, 1992). There is no risk of having lost links, or deleting data without deleting the graphic, or vice-versa;

- **Stronger and more flexible modeling:** as mentioned before, object-oriented modeling brings the physical implementation closer to the logical design. More flexible definition and management of relationships between objects may be available, bringing the logical model even closer to the actual implementation;

- **compact storage:** since the object structure would only contain the information that is currently present, no blanks are stored, and

minimum disk space waste is expected, as well as improvements on the overall performance of the system;

- **User friendliness:** it is much more natural for the user to understand and query data structured with the object-oriented model, because none of the transformations required by the relational model have to be implemented.

On the other hand, it is expected that relational DBMSs should be more comprehensive in their treatment of data security, database recovery, report generation and transaction logging. This is not only because these types of operations are more natural in a relational DBMS, but also because of the longer tradition and development of relational DBMSs.

## THE PRACTICE OF OOGIS

At Prodabel, the data processing company for the city of Belo Horizonte, we have been using APIC, an object-oriented GIS product, since early 1992. This software has been selected in a process that took about a year, in which we looked closely at what was then available in Brazil, and also visited installations of the designated products abroad. In this time, we have come to a reasonable number of conclusions regarding this software, as compared to the conventional GIS market leaders, such as Arc/Info, MGE and GeoVision, and considering the object-orientation theory we mentioned before.

### The Database

APIC's database engine is called *cchr*, which is a French acronym for "coexistence of hierarchical and relational concepts." This means that the database is capable of relating objects in the way a relational and/or a hierarchical DBMS would. To that, it has been added the capability to relate objects in a network, or a graph, as well. Notice that any single object can, potentially, be related to a large number of others, by any combination of the forms of relationship described above.

APIC manages a continuous geographic database. There is no need to divide the space in tiles, or to create indexing maps. Spatial indexing is made by a very efficient meshing algorithm. It is possible to have part of the data residing in an external relational DBMS, and make APIC transparently retrieve the external information when its objects are queried. It is also possible to execute SQL queries from within APIC's programs.

APIC works in a client-server environment. For each defined database there is a server process, that communicates with client processes, which in turn interact with the user. It is possible to have not only several clients to a given server, but also special clients that are capable of talking to several servers at once, in a form of distributed databasing. Object orientation helps in that too, because the volume of traffic over the network is reduced due to the compactness of objects, and also because defining the distributed database's

logical design in terms of objects helps to decide in which database objects will reside.

**Modeling**

Figure 1 shows a partial data model for a water network application. We had to adapt our existing data modeling methodology to create it, because we felt that entity-relationship lacked means to represent several important concepts, like relation type (hierarchy, relation, network) and object nature (graphical, non-graphical). From generic "object-relationship" diagrams, such as the one in figure 1, we create detailed descriptions of objects, determining graphical representation, attributes and relationships (figure 2). We also try to find opportunities for polymorphism, in which we create specialized versions of objects, that will use only a subset of the information that has been defined for the object. These specialized versions can assume different graphical representations. Polymorphism not only simplifies the data representation, but also allows for an easier understanding of the model by the user, in different levels of abstraction.

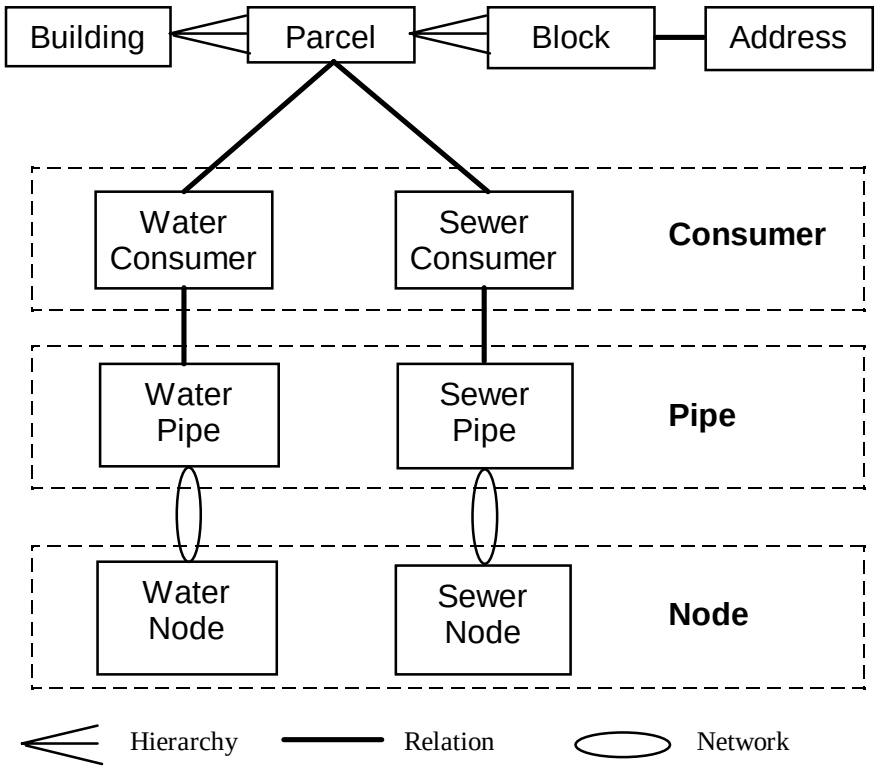


Figure 1 -- Water Network Data Model (partial)

Figure 2 shows the detailing of the water network node object, in which we have defined two different types of nodes: type A corresponds to nodes that interfere with the normal network flow, such as valves, and type B corresponds to non-interfering nodes, such as tees. Either one of the node types will have the same kind of relation with water pipes.

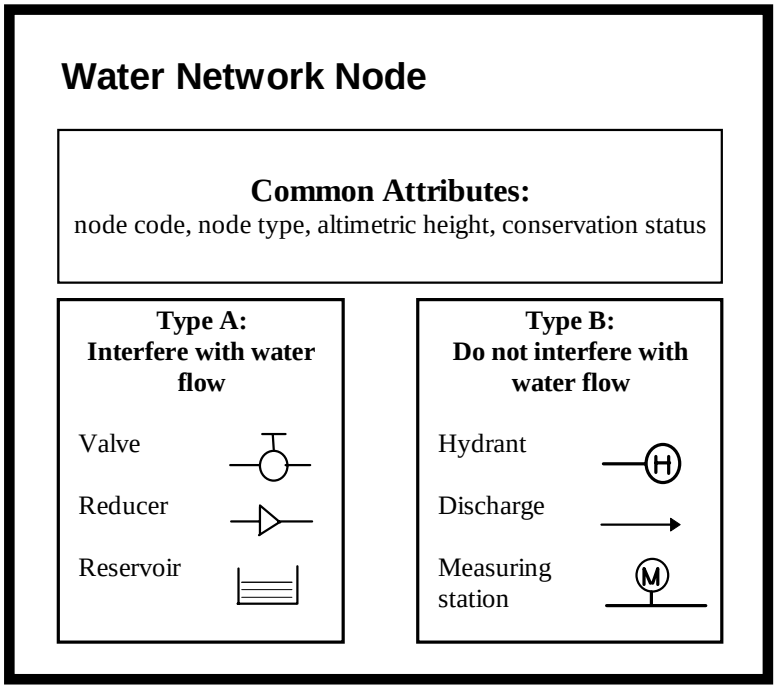


Figure 2 -- Detailing of the water network node model

Objects in APIC can contain graphic and alphanumeric data. It is possible to have non-graphical objects, as well as objects without attributes. The non-graphic objects can be viewed as special "tables," in the relational way of thinking. Nevertheless, they can be related to others also in hierarchical and network modes.

In the definition of each object, which is done in a module called library, it is possible to define macrocommands that will be responsible for data input, automatic relation creation, special display and graphic attribute automatic loading. There are a number of readily available macrocommands, but the user can build his own if necessary. If the user does not choose to modify the associated macrocommands, APIC will provide default methods. There is also the capability to define the range of display scales at which an object will be displayed, as well as special conditions for it to be displayed at all. Table 1 shows the object description for the water network node. In our model, this object can assume several different symbol shapes, depending on the node type attribute. APIC will determine, at display time, which symbol is to be displayed for each node, using the macrocommand AFCOND. Also, depending on the node type, some attributes will or will not be used, without wasting disk space, in a way to implement polymorphism in APIC. The generic node will be related to water pipes, forming a network.

TABLE 1 -- Object description sample

| **OBJECT DESCRIPTION** | |
|---|---|
| **Name** | NOAGUA |
| **Description** | Water Network Node |
| **Nature** | Symbolic |
| **Construction scale** | 1:1000 (4 x 4 mm) |
| **Maximum display scale** | 1:50000 |
| **BEHAVIOUR** | |
| **Type** | NODES |
| **Hierarchical relations** | None |
| **Network relations** | Relation name: AGUANO with type: PIPES |
| **Normal relations** | Relation name: NOAGCX with type: NODES |
| **Confidentiality** | None |
| **Identifier attribute** | CGNOD |
| **Associated macrocommands** | Graphical input: none<br>Automatic relation creation: none<br>Alphanumeric input: none<br>Display: AFCOND noagua.afcond |
| **GENERAL ATTRIBUTES** | |
| **CGNOD** | Node code |
| **TPNOD** | Node type (valve, reducer, pump, reservoir, etc.) |
| **SPECIFIC ATTRIBUTES: node type VALVE** | |
| **DMNOD** | Node diameter |
| **TPRGI** | Valve type |
| **IDESTRGI** | Valve situation: 0 = open; 1 = partially open; 2= closed |
| **SPECIFIC ATTRIBUTES: node type PUMP** | |
| **VZBMB** | Pump throughput |
| **POBMB** | Pump potency |

In the data modeling, APIC supports the concept of *type*, which is a grouping of objects that share the behavior in terms of relationship with other groups of objects. A relationship is not defined between the object classes, but between the types. For instance, consider water network nodes and pipes. There can be different variations of nodes, corresponding to several object classes; likewise, there can be different variations of pipes, and any variation of node can be related to any variation of pipe. Types are a way to have a form

of class hierarchy, though it is not fully implemented as described in earlier sections.

When you try to compare the model for APIC and the normal entity-relationship-relational approach, it is easy to verify that the complexity of the model grows, and also the complexity of the physical implementation. Figure 3 shows the same water network model in entity-relationship, as it would be defined for implementation under a relational GIS.

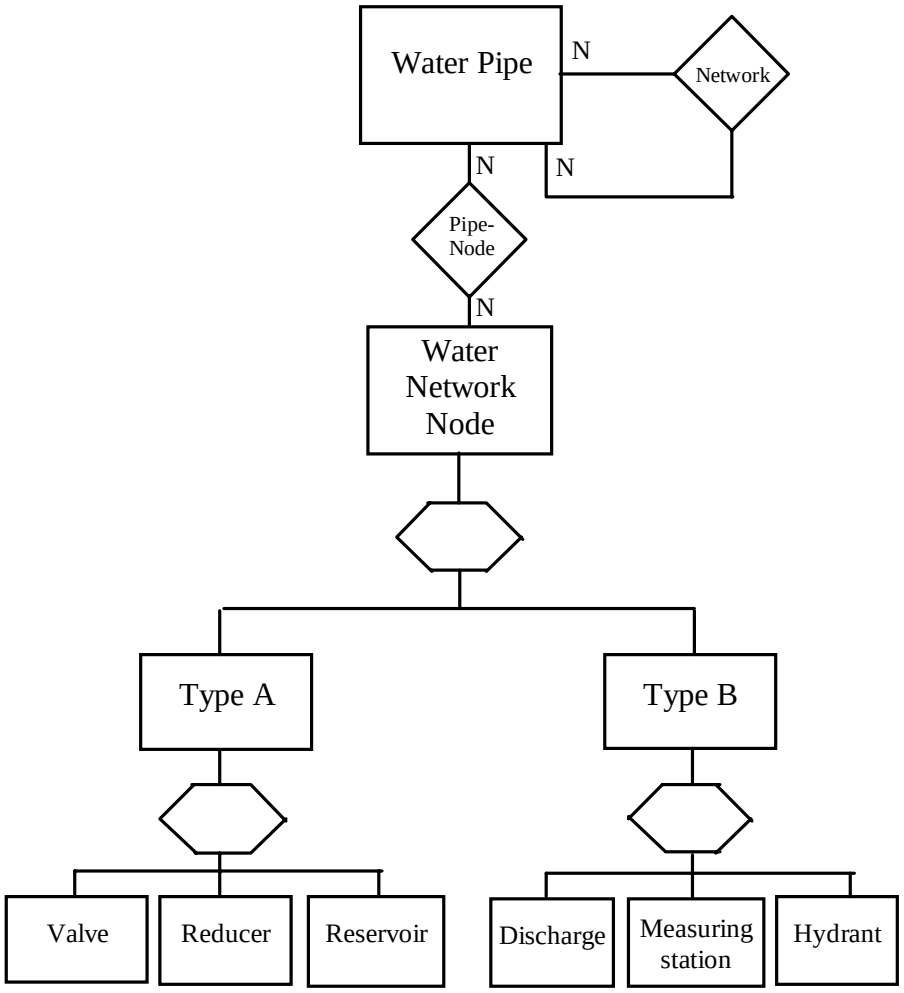


Figure 3 -- Entity-relationship model

A relational schema to fulfill the model in figure 3 would need to have 10 different tables, that is, one for each node type, plus one for generic node data, one for the pipe-node relationship, one for pipe data, and one to represent the network connections. Furthermore, the user would have to know everything about the differences between node types to be able to adequately query the database.

## FUTURE OOGIS IMPLEMENTATIONS

There are a number of good things about APIC, as we mentioned earlier. There are, however, a number of features that either are not implemented in APIC or that we would like to see enhancements for.

Anyway, we think that future object-oriented GIS implementations should have at least these features:

- class hierarchy with inheritance;
- abstract data types definition, storage and retrieval;
- relations between objects in hierarchical, relational and network modes, simultaneously;
- free definition of associated programs, with standard association for inclusion, deletion, modification, display and automatic relation creation programs, plus user-defined functions;
- interactive querying tool, that uses the special types of relations allowed with the object-oriented structure;
- possibility of integration with external relational DBMSs, in a variety of operating environments;
- a structured, modular, object-oriented querying and development language;
- support for polymorphism in the definition of object classes;
- standard transaction control, with data recovery techniques (Edelstein, 1991).

## CONCLUSIONS

After more than two years working with OOGIS, we think that object-orientation is clearly in the path of GIS evolution. Traditional software makers are already seriously considering the integration of object-oriented modules in the already existing packages (Crosbie, 1993), but it will be necessary a full-blown revolution to reach the full OOGIS set of advantages.

Naturally, the pace for the change will be determined by the market's needs and demands. That may take a long time, if users are unwilling to give up the comfortable and well-known realm of traditional GIS, with all its limitations. However, bigger databases and special applications are already beginning to show their weaknesses, since their complexity and response time are approaching the limit of feasibility. At least in our case, with a big 5-million-object database, in over 150 classes, we are sure that it would be

impractical, maybe impossible, to manage the way we do using other technology.

## REFERENCES


Crosbie, Peter 1993 “Reality of Object-Oriented GIS” in *Urban and Regional Information Systems Association Proceedings*, vol. I, pp. 188-199

Edelstein, Herb 1991 “Relational vs. Object-Oriented” in *DBMS*, November 1991, pp. 68-79

Egenhofer, Max J. and Frank, Andrew U. 1992 “Object-Oriented Modeling for GIS” in *URISA Journal*, vol. 4, number 2, pp. 3-19.

Gray, Peter.M.D.; Kulkarni, Krishnarao G. and Paton, Norman.W., 1992 *Object-Oriented Databases: A Semantic Data Model Approach*, Englewood Cliffs, NJ: Prentice/Hall International

Hughes, John G. 1991 *Object-Oriented Databases*, Englewood Cliffs, NJ: Prentice/Hall International

Stone, Christopher M. and Hentchel, David 1990 “Database Wars Revisited” in *BYTE*, October 1990, pp. 233-242

Wang, Joyce C. C. 1992 “GIS Major Data Structures vs. Applications” in Urban and Regional Systems Association Proceedings, vol. II, pp. 13-24.

Worboys, Michael F.; Hearnshaw, Hilary M. and Maguire, David J. 1990 “Object-Oriented Data Modelling for Spatial Databases” in *International Journal of Geographical Information Systems*, vol. 4, number 4, pp. 369-383.


## ACKNOWLEDGEMENTS


The authors wish to thank the National Research Council (CNPq), for granting the necessary funds for the presentation of this paper.

The authors also wish to thank Prodabel's GIS professionals, declaring that this paper is a truly an achievement of the whole team.