The OpenGIS Data Model

John R. Herring

Abstract

The OpenGIS Consortium (OGC) develops implementation specifications to promote the interoperability of geographic information applications. This development is based on the concept of a comprehensive set of common software interfaces supported by geographic servers, across computing platforms. It has begun the process of making geographic information and services an easily used technical information resource. In the process, it is redefining the way many industries look at geographic information.

The Structure and Role of the Data Model

Formally, in the OGC specifications, there are three types of models, each of which plays an important part in understanding the overall approach (see Cook and Daniels (1994)):

- The Essential Models describe how geographic data and applications are viewed in terms of their relationship with "the real world." Each of the volumes of the Abstract Specification contains an English language description that documents this model and, often, the thought process that was used to derive it. An Essential Model answers the question: How ought the world work?
- The Abstract Model is a formal object model that provides implementers with a conceptual core. Originally written using an OMT (object modeling technique) notation, it is being moved into the UML (universal modeling language). Unlike the other models, the Abstract Model must form a cohesive single model that covers all the object types derived from the Essential Model and implemented in the Specifications. An Abstract Model answers the question: What does the software do? Together, the Essential Model and the Abstract Model form the Abstract Specification.
- The Specification Models or, more commonly, the Implementation Specifications each describe how a portion of the overall Abstract Model is implemented within a single or a small number of logically related runtime environments. Currently, Specifications exist for parts of the Abstract Model in COM/OLE, CORBA, and SQL. Work on JAVA and, potentially, XML implementations may lead to additional Specifications. The Specification Model answers the question: What are the objects, and how do they communicate?

The core of this overall plan is the Abstract Model. It ties the implementations to a common view that will allow, at first, data and interprocess communications to cross programming environment boundaries. As Implementation Specifications are created, the Abstract Model is enhanced, creating a feedback process that, hopefully, will eventually evolve into the vision of a common Abstract Model for all implementations (see Figure 1).

The Use of Abstraction and the Role of Polymorphism

The emphasis within OGC on interfaces is derived from the success of object oriented programming (OOP). But understanding OOP is not sufficient to understand the underlying strength of the OGC approach. In the OOP literature, the term "polymorphism" is used repeatedly, but seldom satisfactorily explained or separated from the related concept of "inheritance." Polymorphism is the common usage of interface forms for object classes for common operations. Polymorphism thus implies a "binding" that takes a particular instance of a procedure invocation and "chooses" the appropriate implementation based on the types of both the input and output parameters. Inheritance is the use of an "is a kind of" hierarchy to create common interfaces and use common implementations. Inheritance does produce a kind of polymorphism, but it usually restricts the meaning of the term. For example, inheritance seldom allows you to change the return type of an interface.

This definition is still restrictive, because it only applies to a programming language. The term can be applied across programming environments with some modification. First, we have to separate it from inheritance and other implementation concepts that cannot properly span environments. To do this, we look at behavior as defined by the functions and procedures supported by the system. An abstract protocol is a formal definition of a function or procedure, with specification of input and output parameters (as ADTs, see below) and of the expected behavior. In OGC, UML is used for these definitions, with behavior expressed either in OCL (Object Constraint Language), in predicate calculus, or in English language descriptions. Collections of abstract protocols are abstract data types (ADT). One abstract data type is a subtype of another type if it includes all of the abstract protocols of that other type. The implementation of an ADT in a programming environment is a class. Note that the type hierarchy of ADTs is strictly a set theoretic one, and does not restrict the "inheritance" hierarchy of the classes that implement them. Two classes are isomorphic if they implement precisely the same abstract protocols. One class is homomorphic to the second if the second supports all of the abstract protocols of the first.

The OGC Abstract Model is an attempt at a pure behavioral model, which depends only on polymorphism for its internal consistency. It defines the behavior of abstract entities (abstract data types) by defining functional interfaces (abstract protocols) and describing the results of the processes that those interfaces invoke. The implementation of these behaviors is not dependent on the typing mechanism or on any form of inheritance. Classes that implement these interfaces, even in different programming environments, are functionally isomomorphic to one another, but do not necessarily share any common internal data structures. Simply put, the OGC Abstract Model is based on abstract data types.

This contrasts sharply with a standard based on common data types, which concentrates more on the structure of data than the behavior of entities. The only common data types defined in OGC (the well-known structures) are used for parameter passing between objects.

Server Technologies, Oracle Corporation, President's Plaza, 196 Van Buren Street, Herndon, VA 22070 (jrherrin@us.oracle.com).

Photogrammetric Engineering & Remote Sensing, Vol. 65, No. 5, May 1999, pp. 585–588.

0099-1112/99/6505–585$3.00/0

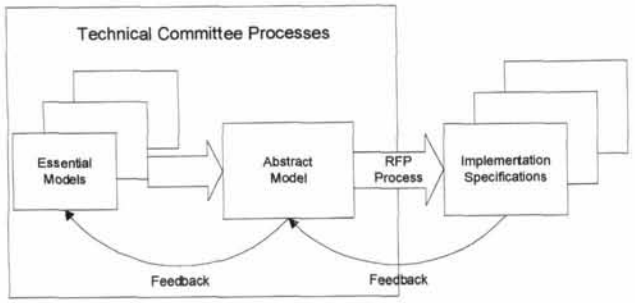

Figure 1. Parts of the OGC model and their basic relationship.

To understand polymorphism in its more general manner, look at mathematical systems. A mathematical system is based on elements, operators, and an algebra of those operators that describes their behavior, especially in the manner in which operations interact. Thus, the algebra of numbers, vectors, matrices, and function spaces all share common look and feel based on commonly defined operators such as addition and multiplication. This is an example of "generic" polymorphism; one based on a common behavior.

The OGC implementation specifications define behavior within each computing environment. The Abstract Specification ties all of these implementation specifications to a common "algebra" of behavior.

Polymorphism and class typing are separable, as can be seen in COM and JAVA, but the terminology used is the same. Even though the collection of protocols that make up a polymorphic interface do not require the concept of type, we use the word "subtype" to describe the objects that behave according to the interface. More properly, we should say that "A supports the interface B," meaning that the implementations of A supports the protocols defined in B and that the algebra of the interfaces, usually expressed as constraints in the test suite, is adhered to by the implementation.

Features: The Root of all Geographic Information

The discussion of polymorphism seems rather abstract, but the concept is the core of the methodology used in building the OpenGIS Abstract Model. The model begins by defining in a very general way what a "feature" is, by defining the way in which it behaves. A feature is any entity that, in addition to other attributes, may have some geographic description, such as a georeferenced geometry object defining its extent. This is a very generic definition, so much so that almost every data item we run into in geographic processing is a feature; as determined by its behavior as opposed to a taxonomic hierarchy (see Figure 2; Figure 2 is a UML diagram).

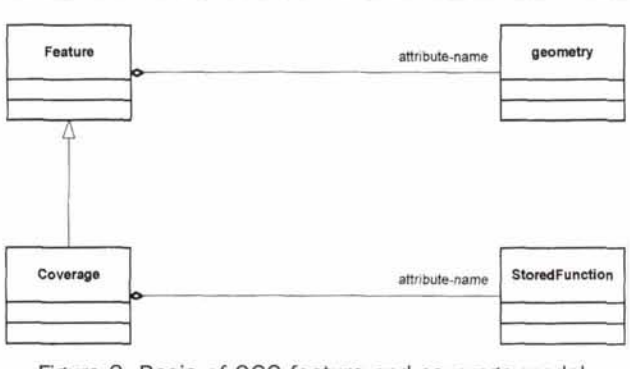

Figure 2. Basis of OGC feature and coverage model.

We are not saying that there is any deep semantic to this, but that the generic definition of feature has led to a set of interfaces, and behaviors, that is quite common to any geographic kernel of data. We are also using a behavioral test for what a feature is. For example, some people find it uncomfortable to consider a georeferenced image to be a feature, but the feature interfaces are valid for it. Thus, in the OGC Model, an image is a type of feature. Similarly, feature collections are features, as are catalog entries, metadata entries, and even (by behavioral definition, but by a little more obtuse route) coordinate reference systems.

The next step in defining the OpenGIS Abstract Model is to define the types of attributes (or properties) that a feature can use for attribute values.

Geometry

The most obvious accomplishment of OGC to date has been the creation of a first generation geometry model which has been agreed upon by its members. The common thread through the three Simple Feature Implementation Specifications (for COM, CORBA, and SQL) is the common geometry model for the description of the spatial extent of feature.

OGC has not been standing still on this. It was recognized that the geometry model for the first implementation specification was limited (the toll of the consensus process), and would have to be extended to meet the needs of more complex geographic applications. Further, this model should agree with (be in "harmony" with) models used in other geographic information standards. The most important example of this is the geographic model used in the SQL/MM (multimedia) type specification that will accompany the advent of the object version of SQL. This query language which is generically called SQL3 (third in the series) will probably be officially called SQL'99 for the year in which it achieves international standard status (through ISO).

The current version of the OGC geometry model is equivalent to the core part of the first version of SQL/MM, Spatial, though this first SQL version also includes circular curves, compound curves, and curve polygons as optional types. A more advanced version of the OGC model is being worked jointly by SQL, ISO TC 211 (Geographic Information and Geomatics), and OGC, and will reach ISO Committee Draft status in 1999. This new model addresses additional interpolation methods, full three-dimensional geometry, and full topological structure options.

The complexity and depth of the new OGC/TC211 geometry model will probably require several intermediate implementation steps before it is fully supported in OGC Implementation Specifications. Although details are changing, the basic approach will remain the same behavioral modeling. The current model, the one used in the "Simple Features" specification, is a 2D (or more precisely, 2½D) cartographic model which uses only linear interpolation and is valid only on a logical surface. The model currently under comment by both OGC and ISO TC 211 extends geometry classes to a full 3D model, and defines a full topology for each dimension. The intent is to supply a very rich model from which "information communities" can create simpler profiles as appropriate to their applications.

The major difference between the older model and the new one is the inclusion of topology. While topology is not uncommon in geographic information systems, this model forces a consistency between the topology primitive and the

geometric primitives that enable the use of computational topology for spatial analysis (see Figure 3).

Coverages: Features that Change over Spatial Extent

Classical cartographic features have spatial extent and other attributes. The interaction of these two types of descriptors is usually ignored. But it is perfectly valid to speak of an attribute varying over the extent of a feature, using a function to describe this (see Figure 2). For example, one might describe the elevation of a surface by using any one of a number of digital elevation models. Images are the most obvious example, because they have a spatial extent (called a "footprint") and an attribute "spectral value" which changes over that extent. Raster images are those with spectral values organized as grids of pixels.

One of the early controversies in the creation of the OGC Abstract Model was what to call these "things that vary." "Image" was the wrong word because it carries the connotation of a visual interpretation. "Field" was suggested and may have been a viable choice, but most disciplines use this to indicate some level of differentiability of the attribute function, which is not the most common type seen in geographic information. The term agreed upon finally was "coverage" for no other reason that it had the least specific technical meaning in common use in geographic disciplines. Or maybe it was the metaphor of a warm and fuzzy cover of attribute draping over the landscape that gave the term its indisputable charm.

Actually, it doesn't really matter what they are called, because the generic feature interfaces do not change when the concept of a "coverage function" is introduced. What does change is that a new attribute behavioral interface is introduced — the "stored function." A function is an association of one value type (the domain type) with another (the range type) where each domain value has a unique range value. This is, of course, the same definition that is used in programming languages, but the difference here is that function-specific code is not normally used to define individual "coverage" functions; rather, a data structure is "stored" that allows generic code to calculate (or interpolate) values for the function. Hence, the name "stored function" to prevent confusion with programming languages functions.

The most common "stored functions" currently in use are images, and other grid data types. These generally share an underlying data structure (a rectangular array of function values), but not necessarily an underlying interpolation scheme. There is currently an OGC RFP (request for proposals) out for submission that will determine the fine points of these types of stored functions, which will soon be followed by another RFP for other types, such as TINs and splines.

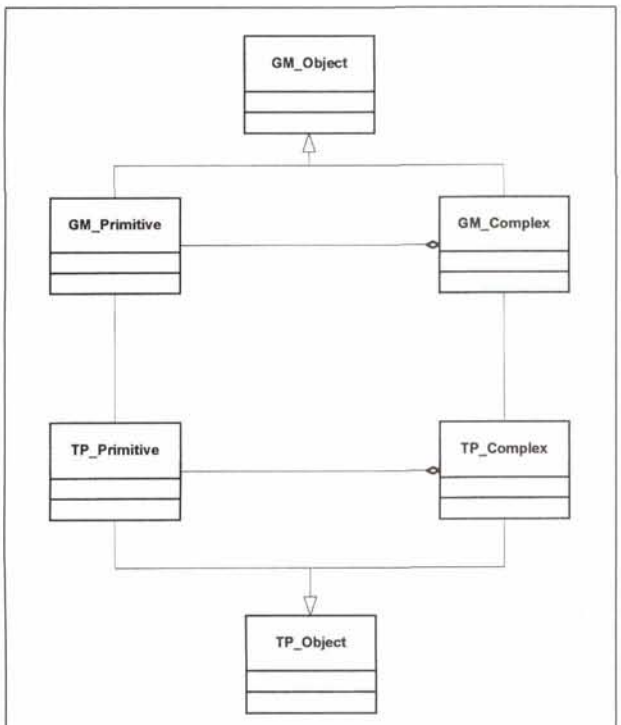

Figure 3. Relation between topology and geometry.

Coordinate Reference Systems

Another arena of cooperation between OGC and ISO TC 211 is in the definition of interfaces to coordinate systems and transformations. A compromise model was accepted by OGC late in 1998, and RFPs for implementation specifications should be coming forward soon.

Catalogs and Metadata: The Third Leg of Geographic Data

Metadata, data describing data, is as important as feature or coverage (image) data to the geographic community. It is the first line that any user will encounter in trying to plan a project based on geographic information. Simply put, it is less expensive to reuse data than to recreate it. Catalogs, in OGC terminology, are applications that access metadata based on user query.

On the other hand, no segment of the geographic information community is more isolated from the OGC behavioral modeling paradigm than metadata. Current metadata available on the network have historically been "human readable" text files full of free text or, in some cases, semi-structured data designed for "librarian" access. In either case, the emphasis has been heavily on data structure and not on interfaces.

This is precisely why the OGC model is as it is. A behavioral model allows different implementations to coexist and to interoperate based on ADT-controlled class homomorphisms (implemented by "bridge code"). Figure 4 illustrates this point. Metadata can be viewed as another form of feature data. Normally, a feature is thought of as a "real-world entity" represented by a "digital object." In the "metadata as feature" metaphor, the "real-world entity" is simply another "digital object" (a feature collection) that is represented by another "digital object" whose attributes are normally called metadata. This recursive use of feature object to describe feature objects and feature collections allows implementations

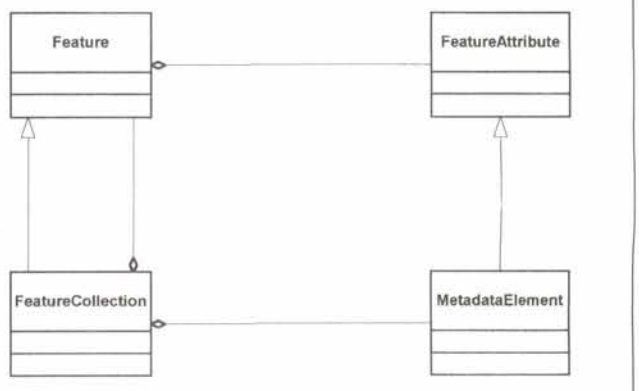

Figure 4. Metadata as features.

to reuse code and user interface. More importantly, it presents a uniform view of "add data" to the user that makes his feature tools and concepts operate as metadata tools and concepts.

This recursive model is not universally accepted, mainly for historical reasons. Years of working with the current text-based systems have led some to expect that a metadata system built for data discovery will necessarily be less "spatially aware" than a data system built for analysis. The simple logic of the recursive design and its obvious validity are sometimes not sufficient to overcome this expectation.

Feature Identity and Relations

The need to find and identify versions of a digital feature has always been a known problem in the OGC community. It has its roots in two major problems:

- The ability to relocate a particular feature so that value-added data can be stored separately from the base feature and not require redundant storage (and its associated integrity problems); and
- The ability to express arbitrary feature-to-feature relations, even across "dataset boundaries."

The Feature Identity and Relation SIG (Special Interest Group) is currently building a behavioral model for this.

Summary

The OpenGIS Interoperability project is centered about the creation of a single common Abstract Model that can be implemented in various geographic information processing environments.

References

Cardelli, Luca, and Peter Wegner, 1985. On Understanding Types, Data Abstraction, and Polymorphism, Computing Surveys, 17(4): 471–522.

Cook, Steve, and John Daniels, 1994. *Designing Object Systems: Object-Oriented Modelling with Syntropy*, Prentice Hall, London, 389 p.

Herring, J.R., 1990. Using Category Theory to Model GIS Applications, *Proceedings of the 4th International Symposium on Spatial Data Handling*, Zurich, Switzerland, 2:820–829.

———, 1991. Using Spline Functions to Represent Distributed Attributes, *Auto-Carto 10: Technical Papers of the 1991 ACSM-ASPRS Annual Convention*, Baltimore, ACSM-ASPRS, 6:46–58.

OpenGIS Specifications, **http://www.opengis.org/techno/specs.htm**.

OpenGIS Guide, **http://www.opengis.org/techno/guide.htm**.

UML (Unified Modeling Language), **http://www.rational.com/uml/index.jtmpl**.