IMPLEMENTATION OF OBJECT ORIENTED GIS USING FORMAL DATA STRUCTURE WITH PLANAR TOPOLOGY - SOME CONSIDERATIONS

Hongguang Yang and Wolfgang Reinhardt, Munich/FRG

Abstract:

This paper is intended to characterize how an object oreinted model for Geographical Information System (GIS) of new generation may be designed and realized. Special emphasis will be laid on the data structure aspect. Design methodes, on the one side used specially for topological structuring of spatial data, and on the other side used generally for object oriented software engineering will be considered here. As a bridge combining the application requirement and software engineering methode the Fomal Data Structure (FDS) from the GIS theory of Molenaar is quite suitable.

It will be assumpted that the reader has been already confronted with topological structure of spatial data and one of the Molenaar's papers about the FDS.

Key Words: Formal data structure,
Object oriented model, Planar topology.

1. Introduction

Nowadays GIS must possess a planar topological structure over its spatial data so that spatial analysis or spatial query of Geographical Information are supported with an acceptable performance, or even possible.

A history GIS which existed before this knowledge is well known shows often the problem that planar topological data structure can hardly re-implemented because at the begin of the system design data topology were not taken into consideration enoughly. Even a re-implementation of planar topological structure is possible, re-structuring of existing enormous data set and re-design of processing algorithms are necessary which would be very work-consuming.

The design and realisation of a new generation GIS is subjected not only to the essential requirement from topological aspect of data structuring, but also from software engineering methode of object oriented design and programming.

There is yet no standard methode to implement planar topology for a GIS to make data and algorithms portable in a meta-level, between different dates (evolutional aspect) and systems (common use of data). In the practice every GIS manage the spatial and non-spatial data at its own way and in different database model (relational or network). The multiple use for changing applications within one GIS and common access of data from differnt GIS are subjected to very strong restrictions.

The planar topological structure of spatial data has taken its standard form, originated from the DIME and succeeded by TIGER (Boudriault 1987). The advantages have been shown in last years contineously. Nevertheless no practicable and widely acceptable methode do exist to make it possible to implement a planar topological structure in an object oriented system environment.

The Formal Data Structure of Molenaar can serve as an meta-language for description of geographical information as a whole, including topological, spatial and non-spatial aspects and independent of used data model. The inherent connection of FDS with object oriented model makes it suitable to standardize the process for GIS design and realisation. To do this, the FDS must be refined and made more understanable.

In this context the following sections will deal with FDS, object oriented model and the connection between them.

2. Characterizing Formal Data Structure

After Molenaar difference must be made between data and information in a GIS. Geographical data is defined in a GIS which is implemented in a computer. But geographical information is resulted from the thinking process of human. We capture geographical data from the reality to a computer by tranforming the geographical information or we get geographical information from a computer by tranforming the geographical data. The interactions between computer and human are 'semantic transformations'. In a wide sense the process such as get a display file from a database is also a semantic transformation.

We are used to think of geographical objects its data form is called 'terrain features' (point, line and area features). Every feature is associated with a 'feature identifier' (Fig.1). Every identified feature can be described by a geometic and a thematic aspect.

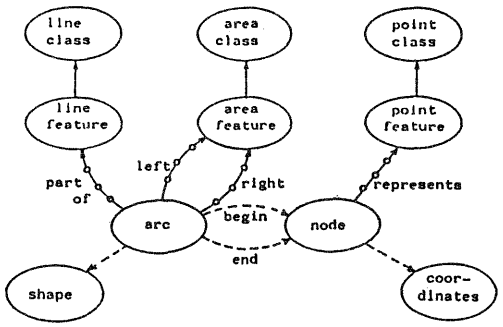

Fig.1 FDS for Single Valued Polygon Map (Molenaar 1989)

Till now two thinking processes can be applied to. The first is classification of features to feature class: 'point feature class', 'line feature class' and 'area feature class'. The second is generalisation of some properties from all features of one class to the feature class itself: since every point feature possesses a pair of coordinates, the point feature class possesses the description with a pair of coordinates. A further generalisation can be carried out if a super class - 'class of terrain feature classes' - is created: since every feature possesses a geometric and a thematic description, the 'class of terrain feature classes' possesses the property being descripted by geometry and thematics.

A class corresponds to an entitiy in the database theory, so that entity-relations may be well defined between classes. However, as will be shown in later, entity-relationships give only data oriented models that is only one aspect of FDS.

The classes of point, line and area features construct the contact level of GIS to human without specifying the way to describing the positions, shapes and topology that are located at a more primitive level.

In this primitive level a class composed of nodes and a class composed of arcs are defined. They construct the complete geometric network with a planar topological structure: non arc has a crossing point with another arc without creating an node; non node overlaps an arc without spliting it; and non node overlaps another node.

Every node possesses a pair of coordinates. Ervery arc (an edge or a non-enclosed curve) is associated with a shape description (may contain more coordinate pairs). Between the two classes entity-relationships must also be pre-defined: every arc begins with a node and ends upon an other node. Between the feature and the primitive level some constant entity-relationships must also be pe-defined: Every point feature is associated with a node; Every line feature is associated with one or more arcs and interior nodes; Every area feature is associated with more arcs.

As consequency all inter-relationships between terrain features are not defined as entity-relations upon the feature classes. They are expressed indirectly by the entity-relations between terrain feature classes and geometric primitives; and entity-relations within primitives themselves. For user only terrain features are accessable. Every thing which occurs at the geometric and topological level is organized automatically by the system.

It should be noted that nodes and arcs as primitives are not accessable to user for topological consistancy. Manipulations (creation, updating and deletion) can only performed for satifying the requirements generated from the manipulations on the feature level. Explicit functions are only defined on the feature level.

For example a spatial query to test if two line features have a crossing point initializes the process to access all arcs by tracing the entity-relationship (line_feature_to_arc) and to find out if the related arcs have a common node.

More interests should be spend to the opposite side of cassification and generalisation, that is the inheritance of 'properties' from super class to classes and from class to features. Not only the 'attribute properties' like the 'description by geometry and thematics' are inheritable. More important are 'functional properties' that will be dealt with in the next sections.

3. Characterizing Object Oriented Model

In the sense of object oriented design and programming (Kim, 1990) every object can be understood as an encapsulaed unit composed of a set of data and a set of functions to operate on the data. Data set (attributes) gives the state of, and function set (methods) the behavior of the object.

The behavior of an object is involved via message passing to it. A message is always composed of three parts: the identifier for obtaining the object to which the message is to be sent; the selector for specifying the state or behavior of the object; and the optional arguments that can be evaluated by the object (they may be attributes, objects or messages). For passing message to an object message interface must be defined for it. There is no way to access an object execpt through the public interface specified for it.

By grouping objects sharing the same set of attributes and methods into a class -it is a new object- the common attributes and methods can be factored out from individual objects into the class (classification). Every object which belongs to a class is an instance which inherits all the attrubutes and methods of the class and may have additioal attributes and methods.

Classes as objects can themselves be grouped into a superclass (generalization). Similiarly all attibutes and methods defined for a superclass are inherited into all its subclasses.

A class may have any number of subclasses. However, a constraint can be made for an object oriented system that a class may belong to only one superclass. In this case a class inherits attributes and methods from only one superclass; this is called single inheritance. Without the constraint a class can inherit attributes and methods from more than one superclass; this is called multiple inheritance. In a system which supports single inheritance, all classes form a hiearchy.

Object oriented model includes the core properties of many well-used models such as entitiy-relationship model and semantic data model. The class concept captures the (instance-of) relationship between an object and the class to which it belongs; the concept of a subclass specifying its superclass captures the generalization (is-a) relationship; and the composition of an object in terms of attributes capture the aggregation relationship.

An significant point of object oriented model is the way dealing with the attributes and their domains. The domain of an attribute is a class to which the values of the attribute belong (e.g., integer, string etc). This means that every integer or string value is an object or instance. The class of integers and the class of strings are only two primitive classes. The domain of an attribute can also be any non-primitive class composed of objects other than alphanumerics. Sometimes a constraint is made that the domain of an attribute should only be a class hierachically rooted at a user-specified class to avoid the nested structures.

Obviously the relationship between an object from one class and the objects from the classes as domains of attributes of the first object gives a class hierarchy, which is called class-composition hierarchy and has nothing to do with inheritance of attributes and methods. The class-composition hiechy corresponds to the aggregation relationship and can be well-used for (e.g.) inter-relationships between the geometric primitives.

A futher important property of object oriented system refers the extensibility: an existing system can be extended without introducing changes to it. The behavior of an object may be extended by simply including additional methods. Besides this predefined attributes and methods of an object may also be re-uesd and inherited into a new specilized object.

4. Implementating Formal Data Structure with Object Oriented Model

The analogy of classes in object oriented model and in FDS is quite evident. Point features from the point class, line features from the line class and area features from the area class correspond to objects or instances. Based on the Fig.1 three levels may be dealt with as follows:

At central level (in the sense of user) terrain features are refered, thus the feature level. Thematic description starts from the feature level upwards and in association with calssifications, whereas geometric description (including shap and planar topology) starts from the feature level downwards.

It should be noted that Fig.1 does not contain classes for composite features. A composite feature has attributes that are themselve feature identifiers from other classes.

An example of composite feature is given as follows. Defining class house which is composed of some area features, the class meadow which is composed of some other area features, and the class footpath composed of some line features. Then the class garden - its every feature with attributes refering an area feature from the class house, an area feature from the class meadow and a line feature from the class footpath - is a class composed of composite features. In this case, house feature, meadow feature and footpath feature are parts of garden feature (aggregation).

Composite features and its classes are important for describing geographical information. They are not considered in the basic figure of FDS explicitly because they vary from one application to another. Based on the extensibility of object oriented model insertion of new classes to an existing class hierachy or network does not effect changes of the exsiting system and thus may be carried out in an any time later.

Special emphasis will be made for the geometric description of terrain features. As shown in Fig.1 links are drawn between node and point feature, between arc and area or line featurre.

The relationship between point and node class is actually a bilateral class-composition hierachy: every point feature possesses an attribut its value reprensents an identifier to a node from the node class, and vice vesa. Note that a node can but not necessaryly possess the meaningful attribut value as identifier to a point feature.

Coordinates are objects of class composed of pairs of real numbers. The relationship between node and coordinate pair leads to another class-composition hierachy, not necessaryly bilateral.

But more important are here functions or methods associated with objects and classes: by defining the point feature class, functions may be defined simultaneously that must be carried out in association with messages passing to, for example in the process of creation, updating and deleting of a point feature.

For example, two functions:

Fp1=(send a message to node class for starting the function Fn, the message contains the identifier and the coordinate pair of new created point feature);

Fp2=(write the value of the passed identifier to the attribute of actual point feature for refering a node object)

are associated with point feature class. Another function:

Fn=(find the existing node or create a new node with the given coordinate pair passed by and send a message to the point feature with passed identifier and start the function Fp2, the message contain the found or new created node identifier)

is associated with the node class.

The creation function of a new point feature associated with the point feature class starts the function Fp1 immediately after a new point feature is created. By tracing the way of message passing the new point feature get the reference to its node.

The relationship between area feature and arc is also a class-composition hierachy and bilateral aggregations: every area feature has a set of attributes their values refer arc objects from arc class for representing the contours of an area. Conversely every arc has two attributes their values refer two areas features (left and right area) from area class. Methods associated with area feature may be more complicated. Because a series of geometric-topological conditions must be satisfied for arcs such that they construct a area feature. It is not possible to specify them without more detailed specifications.

Other relationships between the feature level and the geometric level can be refined at the same way which will be not treated here in more detail.

The consistency of planar topological structure within geometric data and hence correct inter-relationships between the terrain features can only be guaranted if manipulations of geometric-topological level are involved by messages exchanged between feature level and primitive level. Of course powerful functions for handling geometric data and topology are required.

5. Concluding Remarks

Conclusively said FDS can serve as object oriented data model of geographical information in a meta-level, which integrates spatial (geometric) data with its planar toplogical structure, nonspatial data and inter-relationships between the spatial and nonspatial data in a whole system.

To implement a new-generation GIS with object oriented model FDS can be well-used but it must be refined in two aspects: on the one side, a more detailed meta-structure is required for defining thematic, geometric and topological data that will be evaluated by applications (state model); and on the other side, the behaviors of object features and classes (functions or methods) must be specified for implementation (behavior model).

Besides this, object oriented programming languages such as C++, Object LISP etc play an important role because they

support the implementation of object oriented model in an efficient way and can establish standard methods for GIS software engineering.

References:

Boudriault, G., 1987. Topology in the TIGER File. Auto Carto 8 Proceedings, pp.258-263.

Kim, W., 1990. Introduction to Object-Oriented Databases. The MIT Press Cambridge, Massachusetts.

Molenaar, M., 1989. Single Valued Vector Maps - A Concept in Geographic Information Systems. Geo-Informations-System, Vol.2, No.1, pp.18-26.

Adress of Authors:

Dr. H. Yang/Dr. W. Reinhardt
Siemens Nixdorf Informationssystem AG
Fachzentrum Geo-Informationssystem
Otto-Hahn-Ring 6
D-8000 München 83