HANDLING INTEGRITY CONSTRAINTS OF COMPLEX OBJECTS IN SPATIAL DATABASES

Hanna H. Kemppainen
Researcher
Finnish Geodetic Institute, Finland

Commission III

ABSTRACT

The study reported in this paper is based on two assumptions about the importance of the consistency of a geographical database. Firstly, consistency is an important factor concerning the quality of data. Secondly, consistency of a geographical object requires that the description of the context of the object is a part of the object definition.

In object-oriented databases, it is possible to model geographical entities as complex object types whose semantics are captured through constraints on objects and object relationships.This paper examines the modelling of geographical objects and spatial relationships between them. A geometrical object model for geographical entities is presented which consists of 1) the structural definition of the objects, and 2) the enforcement of the implicit structural constraints as object methods. A modelling technique for specifying explicit constraints on spatial relationships between objects is also presented.

KEY WORDS: Geographical modelling, Integrity constraints, Object-oriented databases

1 INTRODUCTION

1.1 Motivation

Consistency of a geographical database is an important factor of quality of geographical data. The description of quality of data provides information for a user to evaluate the fitness of the data for a particular use. This study is motivated by the lack of support for user-defined constraints in commercial GIS software. As to the author's knowledge, there are no such systems where the user might specify arbitrary constraints on the database. The purpose of this paper is to present issues related to the consistency of the data in the database, what is the meaning of constraints of the database and what the constraints on geographical database might be. These issues are handled in order to show what could be required from a software system that manages a geographical database as far as the consistency of the data is concerned.

The specification and enforcement of integrity constraints in object-oriented databases is studied also in this study. This is done to show the possibilities of high level data models to capture semantics of data, which capability is lacking from conventional, record based data models. The significance of high level data modelling lies in the understandability of the concepts to the end-user of the system, and hopefully also as a programming work simplifying agent.

A database is a representation of some portion of the real world, according to the requirements of the application that the database is going to support with its data. The requirements for data are specified in the database design phase, and they are formalized in the database schema. The database schema is an instance of some data model. The database schema is the definition of the correct, allowable types of data, whose instances may be stored in a particular database. The schema thus constrains the instances of data types to some pre-defined structure. The requirements that cannot be presented in the schema are presented outside the schema, for example, in application programs. The contents of the database is constrained in much the same way by these two types of constraints; only their specification is different.

Consistency is probably the most important criterion for the use of the database once its usability has otherwise been demonstrated. As a database serves as an information-providing element in an information system, the user of the database has to be sure that he can trust the image of the real world the database is offering. Therefore, the user of the database has to know the semantics of the constraints put on the database and the processes by which the constraints are enforced; he also has to understand the results of the constraint enforcement.

An example of an integrity constraint referring specifically to a geographical database is that a location occupied by an object representing a house may not be occupied by an object representing a lake. Integrity enforcement automatically relieves the user from checking the database by hand, or rather, by eye, that every lake in the database is free from houses. The specification of integrity constraints is application dependent; it is the aimed use of the database that determines the constrained objects and the way they are constrained.

1.2 Previous studies and outline of this study

The importance of consistency is emphasized in for example (NCDCDS, 1987, part III, pp.5-6). Description of consistency tests is proposed as a part of the quality report of the data. The description of consistency is thus a part of metadata describing a geographical database. In (White, 1984) questions regarding the consistency of a multipurpose geographic data system are represented. In (Laurini & Milleret-Raffort, 1991), correct geometric construction of spatial objects is emphasized, and in (Molenaar, 1991) rules are listed which can be used to check the consistency of a database.

In this study, emphasis is on object-oriented modelling of geographical entities, that is, definition of geographical object types in object-oriented databases, and enforcement of their integrity. Geometrical object model is presented which can be used in describing the geometrical structure of real world entities. The structural features of the object model capture the geometry of an object, while the procedural features of the model enforce the structural integrity of an object. Additionally, a technique is proposed by which user defined constraints may be specified to a geographical database.

The concepts of object-oriented programming and databases are not explained in this paper. The author things that the concepts such as class, attribute, method, inheritance etc. are generally quite well known. An earlier paper by author (Kemppainen & Sarjakoski, 1990) is a review of concepts of object-oriented programming and databases, and their application to the development of geographical information systems.

In the next section (2), the relationship between integrity constraints and OO data modelling is discussed in general. Section 3 briefly reviews geographical modelling and object-oriented modelling. Section 4 gives an object-oriented data model for describing the geometrical structure of geographical entities. Section 5 is a proposition for geographical data modelling based on the model of section 4, along with a study of a possible integrity constraint in geographical application. The ideas of this paper are summarized and the future research is outlined in section 6.

1.3 Some definitions

The use of concepts such as geographical information and geographical database, spatial database, geographical and spatial objects, etc. is not always very consistent in the litterature. The meaning of some of these concepts is explained next, the way they are used *in this paper*.

Geographical database is a data storage, in which data about real world entities is stored. The two concepts, 'geographical database' and 'spatial database' are used interchangeably in this paper. Geographical entity is a real world entity, which is represented by a geographical object class in the database. The notion of geographical modelling is understood here as a process of describing geographical entities by describing their geometrical and thematic properties.

2 INTEGRITY CONSTRAINTS OF A DATA MODEL

By database semantic integrity is meant the techniques used to keep the database in a consistent state with respect to the constraints specified on the database (Elmasri & Navathe, 1989, p.589). In order to prevent an inconsistent database state, whenever an update is applied to the database, the semantic integrity checking part of the system determines if the update will cause a constraint violation. The term integrity is used to cover semantic integrity and transaction integrity. Semantic integrity emphasizes the meaning of the integrity constraint, while transction integrity is related to concurrency control and database recovery techniques. This paper focuses on the semantic interpretation of the concept.

Integrity constraints constitute a set of rules that say what the allowable states of individual objects in the database are, and thus what the allowable states of the whole database are. In a database system, the specification of data types is presented in the database schema as a description of the data content in the database. This specification of data in itself constrains the database: when an update operation is performed in the database, be it insert, delete or change of some instance of a data type, the instance is constrained to be correct according to the data definition. This is implicit integrity constraint of a data model, i.e. constraint that is included in the schema. For example, when in a relational schema some relation attribute is specified as a key attribute, a check is made during database update operations that no two tuples having the same key attribute value are inserted into the database. Other example of implicit relational model constraints is referential integrity, i.e. tuple value that is not stored in the database cannot be referred to. Unfortunately, many of the relational model implementations do not enforce this constraint according to (Elmasri & Navathe, 1989, p.145).

What is not implicit in the data model has to be specified explicitly, outside the database schema. In other words, what you can't specify using the schema constructs, you have to program yourself. For example, in the information system built up above a relational database the before mentioned referential integrity constraints have to be enforced in the application program. The semantics of the data is thus embedded in code that is not part of the database, meaning that it is not managed, that is protected from inconsistencies, as other data on a real-world entity. The problem is that several different integrity constraints might be specified to the database, and it is up to the application programmer to be sure that no two conflicting constraints are specified.

The inability of the relational data model to represent semantics on data is one reason why other ways of modelling geographical, and other highly structured, data types have been considered during the last ten years. What is needed in application areas of this kind are data models which support higher level concepts than a simple relation. Higher level data models represent more types of constraints implicitly than lower-level data models (Elmasri & Navathe, 1989, p.596). Thus the semantics of data are embedded in the database schema, not in the application program handling the data in the database.

The technique of object-oriented (OO) data modelling is based on the concept of object, coming from object-oriented programming (Booch, 1986, Kim, 1990, pp.7-11). With a single object, it is possible to present both structural and procedural data about a real world entity. Object class definition presents an abstraction which has been made about a real world. The abstraction might be an association or an aggregation of classifications made before (Sowa, 1984, pp. 103-123, Nyerges, 1991, pp.76-82). Other abstraction techniques are generalization and specialization, which broaden or lessen (respectively) the meaning of an existing abstraction. These are the techniques for getting more semantics into the database schema, and which simplify the application programming. An object-oriented model is implicitly a constrained description of a real world, making it easier to understand the semantics of data.

This does not mean, however, that the problem of data modelling can be forgotten. OO databases offer modelling concepts that are richer in semantic content than those of relational model, and the specification of data types might be more convenient, but the underlying problem still remains: What is the geographical knowledge that has to be captured in a particular problem area, that is, what are the geographical objects.

3 OBJECT-ORIENTED GEOGRAPHICAL DATA MODELLING

What can be required of a geographical data model? It should automatically provide answers to questions that can be answered by visual inspection of a map image (White, 1984, p.16). This requirement implies that the data model should provide concepts similar to the user's experiences of the world (Mark & Frank, 1989), such as object, collection, part-whole, link, containing, near-far, etc. Geographical data modelling is a process whereby these experiental model concepts are mapped as mathematically defined concepts.

The experiences described above can be formally modelled using concepts of geometry. The geometry of a map requires three geometrical base objects (0-cells, 1-cells, 2-cells), two relations (0-1 incidence, 1-2 insidence), and metrical descriptions of the objects (coordinates and shape) (White, 1984, p.16). The topological portion of a map geometry consists of base objects and incidence relations between them. The insidence relations constrain the touching of base objects and, further, provide concepts with which the relationships between the base objects can be described. The concepts of the topological interior and boundary of a geometrical base object are derived from the incidence relation between the objects. For example, the boundary of a 2-cell is defined as a set of incident 1-cells to the 2-cell. The mathematical formulation of the topological interior and boundary concepts has been discussed in (Egenhofer & Franzosa, 1991). The geometrical view of geographical modelling emphasizes the topological properties of geographical entities; the coordinates and shape of the base objects are metrical properties of these objects.

Geographical data modelling is a process whereby an abstraction is made of a real world entity so that the geometrical properties of the entity are emphasized, and semantics are given to the geometrical abstraction by naming the abstraction and characterizing it with attributes. The neighbourhood of an entity, i.e. associated entities are modelled using concepts that emphasize the meaning of the neighbourhood. For example, neighbourhood inferred from the geometry of an entity might be modelled using the incidence relation, while other kinds of neighborhoods might be modelled as sets or tuples.

Object-oriented geographical data modelling is defined here as a description of some portion of geographical reality using the concept of an object, as defined in the preceding section. Thus a geographical entity, along with its neighbourhood entities, can be described as a single object, resembling the conceptualizations humans make about reality.

4 AN OBJECT-ORIENTED DATA MODEL FOR SPATIAL DATA

This section gives an object-oriented data model for describing the geometrical structure of geographical entities. The model emphasizes the geometrical properties of geographical entities. Certain geometrical object classes are defined, which are then used as modelling primitives for geographical object types. The geometrical objects are independent of any application domain, and describing only the geometry of a geographical object. For example, *line string* is a geometrical object class, which can be used to define the geometry of a geographical object, say, a *river*, or *a boundary line of a river*. Geometrical object classes are structural primitives of the model. Spatial relationships between geometrical objects are modelled using the concepts of topological interior and boundary.

The model also includes operations on object classes, which change the structure of an object or derive a new object from an existing object. These operations are called object methods. For example, the method named *boundary* defined for a particular object class is a procedure that extracts the topological boundary of an object.

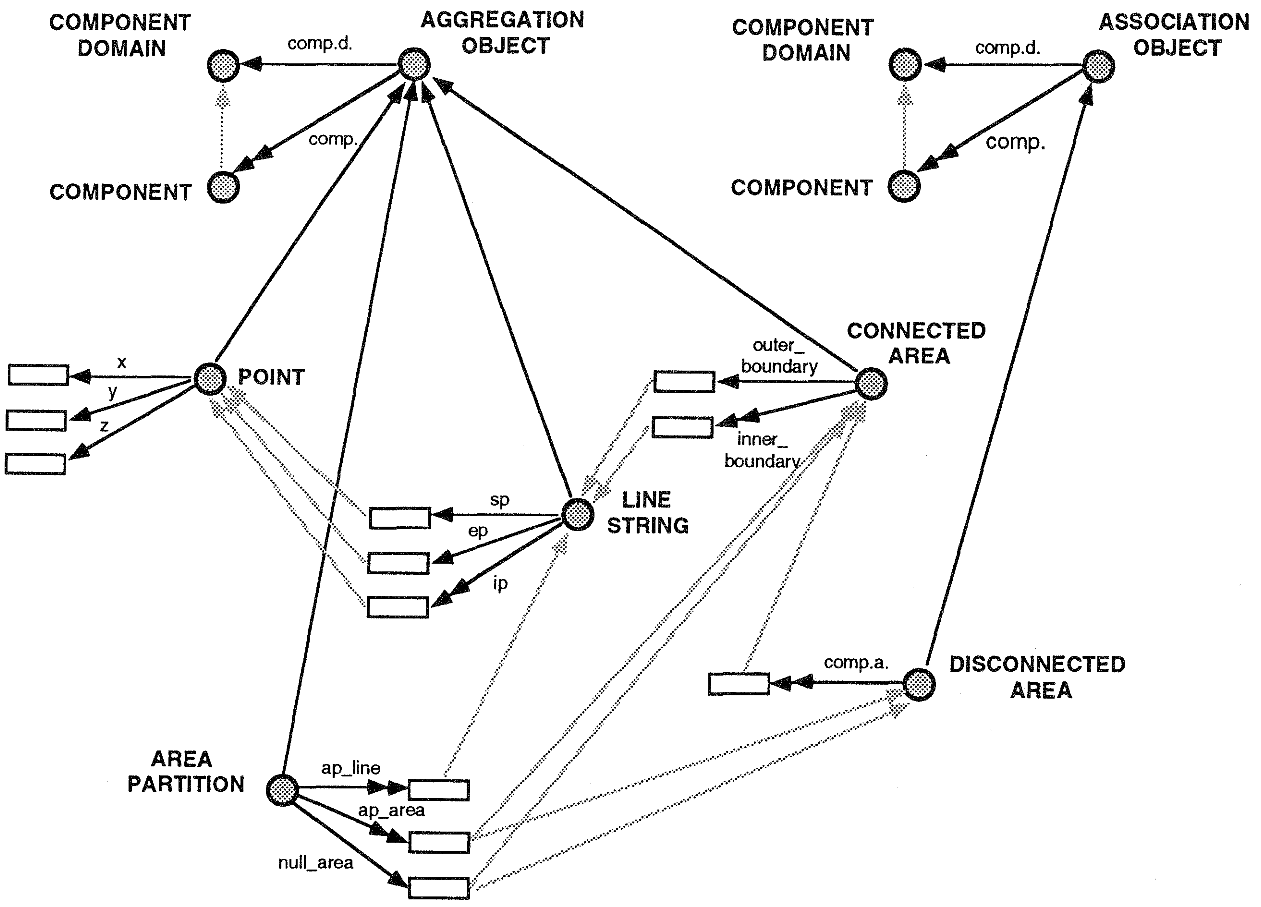

Figure 1. Geometrical object classes.

As mentioned before, the concepts of the model, or modelling primitives, and operations on them are chosen so that they do not imply any semantic content which cannot be described formally. The semantics of objects are captured in the application specific database schema by naming the geometrical objects and characterizing of them with attributes. It is up to the database designer to decide which concepts of the model he wants to use, depending on the processing requirements of the application. For example, geographical reality could be modelled as a collection of isolated geometrical objects, with no spatial relationships between them, or the relationships could be taken into the model, thus permitting information on the context of an entity to be captured.

4.1 STRUCTURAL FEATURES OF THE MODEL - OBJECT CLASSES

The geometrical object model is represented graphically in Figure 1. In the figure, only the structural features of the model are shown, the operations, that is, methods of object classes are described in a later section. The following conventions are used in the figure. Circles represent abstract classes, and rectangles represent stored attribute values. The ISA relationships between objects are shown by unlabelled arrows. Labelled arrows denote attributes of an object class. Multivalued attributes have double arrowheads. As far as the instantiation of the model is concerned, note the two types of values of an attribute: the value may be stored as a simple value (rectangle), or it may be another object, that is, its identification. Grey arrows indicate the domain of the attributes.

The model consists of abstract object classes, and operations, that is, methods defined for them. There are two types of classes in the model 1) Classes that serve for definitional purposes, which by themselves do not describe the geometry of an object. These classes provide a general abstraction mechanism. AssociationObject and AggregationObject are such classes. 2) Geometrical objects, that are subclasses of the classes mentioned above.

OBJECT	CONSTRAINTS ON AN OBJECT
LINE STRING	not intersecting itself
CONNECTED AREA	area boundaries are closing line strings, not overlapping, and with ordering property
DISCONNECTED AREA	component areas non-overlapping
AREA PARTITION	set of non-overlapping areas with covering property

Figure 2. Additional constraints on some geometrical objects

AssociationObject and AggregationObject are both aggregations of ComponentDomain and Component. The Component attribute of each classes is constrained by the ComponentDomain, in which the possible components of the class are listed. ComponentDomain provides a grouping mechanism for the objects that are the only possible building blocks of the aggregate.

Geometrical objects of the model consist of geometrical base objects and geometrical complex objects. The geometrical base object classes are Point, LineString and ConnectedArea. Higher level geometrical objects, which are called geometrical complex objects, are formed by aggregating or associating the base objects. The geometrical base objects are aggregates themselves, for example, point is an aggregate of two or three coordinate values forming a tuple. In this modelling scheme, a distinction is made between the base objects and the complex objects is made to give an idea of the logical connection of this model to the geographical modelling conventions in general. Geometrical complex objects of the model are DisconnectedArea, AreaAssociation, LineAssociation, and AreaPartition.

The object classes are described by their attributes in Figure 1. Figure 2 gives a listing of the implicit constraints on the object classes that are not included in the Figure 1. The implicit constraints on the object classes determine the implicit integrity of the object. For example, the boundary line string(s) of a connected area is constrained to be closing. On the other hand, the boundary line string is an instance of the LineString class whose instances are constrained so that they may not intersect themselves. In this case, the constraint of the component class propagates to the owner aggregate.

The most complex object in the model is the AreaPartition object, whose components are areas and line strings. The semantics of AreaPartition lies in the covering property of the object, so that the subareas, that is, areas constituting the area partition may not overlap each other. This geometrical object type is used to define the geometry of a geographical phenomenon that is known to exist allover the geographical region to be modelled, for example, land use, or real estate division. Note the similarity of this geometrical object type with the consistency principle of a map database in general, presented in (White, 1984).

4.2 OPERATIONAL FEATURES OF THE MODEL - OBJECT METHODS

The implicit integrity of the geometrical object model described above is enforced by the methods of object classes. The methods consist of procedures that determine whether or not the state of an object is consistent with respect to the class definition; the procedure can also *affect* the state of an object so that it will reach the consistent state. For example, the consistency of an instance of the ConnectedArea class is enforced by a defining a method for the class that determines whether or not the boundary of a particular instance is closing.

Constraint analysis is a possible design process for the object-oriented database design environment (Urban & Delcambre, 1990). In the process, the constraints on object classes are first specified using first-order logic representation, which is then converted to Horn clause form. The Horn clauses are numbered, and they are arranged to a graph based the dominating relationship between two clauses. The authors of the paper mentioned above propose that the constraint graph can be used as a help when specifying methods for object classes.

In this study, the analysis of object methods is based on the idea of constraint analysis, but the handling of the problem is different. The analysis of object methods is performed based on the figure 3 which is a description of order between geometrical objects. The semantics of order between objects are the following. If an object class is defined as an aggregate (or an association) of other object classes, the aggregate (association) follows its component classes in the orderscheme. In figure 3, object B follows object A, if object A is in the head of the arrow connecting the two objects. For example, LineString follows Point, because the geometry of a line string is determined by points. Thus, LineString class should be provided by a method, that checks that the components of a line string instance do exist (as instances of the ComponentDomain of ConnectedArea class). Methods for the object classes are listed in Figure 4 based on the structural features of the model (Figure 1) and the order between the objects as in (Figure 3).

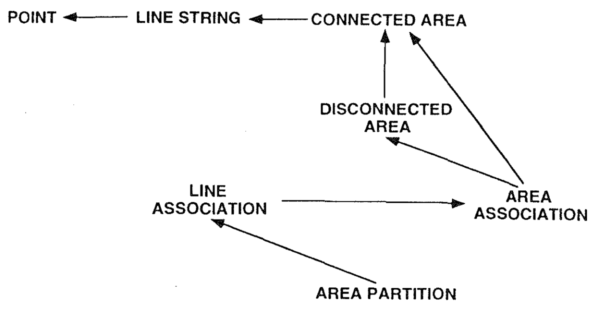

Figure 3. Order between geometrical object classes.

OBJECT	METHOD	METHOD INPUT	METHOD ACTION	ATTRIBUTE THAT GETS VALUE BY THE METHOD
Association Object	CheckComponent	ComponentDomain Component	Check that component is member of ComponentDomain	-
Aggregation Object	CheckComponent	ComponentDomain Component	Check that component is member of ComponentDomain	-
Point	XYZ	coordinate values	Assign values to attributes	x, y, z
LineString	StartPoint EndPoint IntermediatePoint IntersectsItself	Set of points (association of points)	Assign values to attributes Check that line string does not intersect itself	sp, ep, ip
Connected Area	ClosingBoundary OuterBoundary InnerBoundary BoundariesIntersect	Instance(s) of LineString class	Determine order between boundaries If more than one boundary, Check that no two boundaries intersect	outer_boundary inner_boundary
Disconnected Area	AreasDisjoint	Instances of ConnectedArea class	check that areas are distinct	component_area
AreaPartition	NeighborsForLineString NullArea OnePiece	Instances of ConnectedArea or DisconnectedArea class	each ap_line is associated with two ap_areas; one null area exists; one outer boundary;	ap_area ap_line null_area

Figure 4. The methods that enforce the implicit integrity of object classes.

4.3 SPECIFICATION OF SOME EXPLICIT CONSTRAINTS ON GEOMETRICAL OBJECT TYPES

The geometrical object model that is represented in this paper consists of geometrical object classes along with methods that enforce the structural integrity of the classes. The model depicted in Figure 1 is static in the sense that it does not provide modelling constructs that make it possible to constrain the spatial relationships between the geometrical object types. This study is based on the assumption, that in a geographical database it should be possible to specify user defined, arbitrary constraints on the object classes. The fact that some constraints on geographical objects are heavily application dependent, has the following consequences: It is the application database designer who knows best what the constraints on database should be. These constraints don't need to be, or rather, they should not be included implicitly in the data model, because all of them might not be needed in an application, and thus their enforcement would cause unnecessary processing overhead. Rather, constraints that are typical to a certain application should be specified as explicit constraints.

Next, the model depicted in Figure 1 is extended by modelling constructs that make it possible to specify explicitly user-defined constraints on object classes whose semantics are derived from the spatial relationships between objects. Such modelling constructs could be *modelling methods* of the geometrical object model. Modelling method is a metalevel concept in the model.

The modelling methods of the geometrical base object classes are named Boundary and Interior. The action of these methods is to extract the topological boundary and interior of the instance of the class. Spatial relationships between geometrical base objects can be described formally with these concepts. The specification of binary spatial relations is given in (Egenhofer & Franzosa, 1991). According to them, the binary spatial relationship between two point sets can be determined by examining the intersection of the topological boundary and interior of these two point sets. In the paper mentioned above, the authors formalize the concepts that describe binary spatial relationships between two area objects, such as disjoint, touch, overlap, equals, inside, covered by, overlap with disjoint boundaries, and overlap with intersecting boundaries. Formal definitions for spatial relations are a starting point for the definition of arbitrary spatial relationships between *any* two geometrical objects. What is still needed is the development of the theories of the previous study to cover higher level geometrical object types, such as area partition proposed in this study.

The action of the Boundary and the Interior methods is to extract the topological boundary and interior of the object class for which the methods have been defined. The binary spatial relationship between two object types is defined by the boundary and interior points sets of a particular class. An example might clarify the use of this approach: A user of the system might define a constraint on two different area types by help of the Boundary and Interior methods. Suppose that these two areas represent area type A and area type B. A constraint could be specified, whose semantics is the following: area type A must be disjoint from area type B. The method IsDistinct might enforce this explicit constraint: the input to the method are to instances of (for example) ConnectedArea class, and the method action is to determine whether or not an instance of a LineString class is shared by the two ConnectedArea instances.

The constraints of this kind can be enforced by the sequencing of methods of geometrical base object classes given in Figure 5.

OBJECT	OUTPUT FROM THE BOUNDARY METHOD	OUTPUT FROM THE INTERIOR METHOD
Point		Point itself
LineString	Start point, End point	Line string itself except its start point and end point
Connected Area	Bounding line strings	Area itself except the boundary line strings

Figure 5. Description from the output of the modelling methods. The specification of output is based on (Egenhofer & Franzosa, 1991).

5 GEOGRAPHICAL MODELLING BASED ON THE ABOVE MODEL

On the basis of the model of geometrical objects defined above, the following is proposed concerning the geographical modelling for object-oriented databases:

1) The geometrical structure of a geographical entity is modelled using the geometrical object definitions above, and other structural characteristics are described by additional attributes. The result of this modelling step is a geographical object class definition.

2) The neighbourhood of a geographical entity is modelled by forming associations and aggregations of geographical objects. Associations and aggregations are defined using the concepts of the topological boundary and interior of the geometrical objects underlying the geographical ones. The result of this processing step is the definition of a named neighbourhood of a geographical object class.

The two definitions are then used in the implementation of the system. An example of the proposed modelling scheme is represented next.

5.1 Example of geographical modelling using the geometrical object model

The process of geographical modelling based on the geometrical object model presented in this paper is described using a simple example. Suppose that an application needs information on the areal extents of three land use classes, viz. mainlands, islands and water areas, in some study area. The dimensions of the study area are clearly defined, for example, by geographical coordinates, and it has been decided that the boundaries of the land use classes are digitized from topographic maps on a certain scale, say, 1:100 000. It is now the job of the database designer to design an appropriate data model for this application, so that the information on the land use classes is correct with respect to the geographical reality of the map sheets to be digitized.

On the basis of the semantics of the land use classes, the following constraints are put on database:
1) The individual land use classes are represented as connected or disconnected areas.
2) The three land use classes may not overlap.
3) Every location in the study area must be occupied by some land use class depicting area.
4) An area object representing mainland cannot touch (see section 4.3), that is, be the neighbour of an area object representing an island.

The analysis of these constraints, and the geometrical object model represented in this paper leads to the following choices in the schema design. The study area geometry is defined as a reference to the AreaPartition class, because of the constraint of non-overlapping areas, which cover the whole study area. Constraints 1 through 3 are implicit in the geometrical object model.

Constraint 4 has to be specified as an explicit constraint in the schema, by defining it using the Boundary and Interior methods of the ConnectedArea class.

Constraint 4 can be enforced by defining a method FindNeighbor for the Mainland class using the Boundary and Interior methods defined for the ConnectedArea class. The neighboring areas for Mainland are modelled as a MainlandNeighbors class, which is defined as a subclass of the AssociationObject. The MainlandNeighbors inherits the properties of its superclass. ComponentDomain of Mainland-Neighbors is a set of bindings to the instances of Mainland class, whose instances share common LineString instance with the Mainland instance under inspection. The last step in the enforcement of constraint 4 is to iterate through the instances of MainlandNeighbors, and check that no instances of Island class belong to that set.

A list of actions made by the database designer, or database user, creating a database of the above example is given next. The list is based on the assumption, that a software system exists, which is an implementation of the structural and procedural features of the model of figure 1. If such an implementation exists, the object classes of Figure 1 and their methods listed in Figures 4 and 5 can be used as modelling constructs.

1. Define Mainland class and Water class as of ConnectedArea type. Define Island class as of DisconnectedArea type.
2. Define StudyArea as of AreaPartition type.
3. Store Mainland, Water and Island class names as the ComponentDomain of StudyArea.
4. After the digitization, enforce the integrity of Study-Area instance by activating the methods NeighborsForLineString, NullArea and OnePiece (see Figure 4). Result of this step is the values for the ap_area, ap_line and null_area attributes of the StudyArea instance.
5. Define FindNeighbor method for Mainland class. Define the method by the Boundary method.
6. For each ap_area belonging to the Mainland class, create MainlandNeighbors class. Send FindNeighbor message to Mainland instance. ComponentDomain of MainlandNeighbors gets its value from the execution of the method.
7. Iterate each Component in ComponentDomain of MainlandNeighbor, and check that Component does not belong to the Island class.

6 DISCUSSION AND CONCLUSION

A geometrical object model for the definition of the geometry of geographical entities has been given in this paper. With this model, the author has tried to demonstrate two things. Firstly, what are the possible object definitions of geographical entities, when the concept of object is understood in object-oriented programming and databases. Secondly, the enforcement of implicit and explicit integrity of the object classes is studied. The author proposes that the integrity enforcement of object classes is put into effect by object methods.

A model of this kind offers a way to describe complex geographical phenomena using a few well-defined concepts, instead of being compelled to code the semantics of phenomena in application programs. It is also proposed in this paper, that the user should be provided by modelling constructs, that can be used in specifying explicit constraints on spatial relationships between objects.

The ideas of this paper have developed from the author's interest in object-oriented modelling in general, and also from the inability of existing commercial software to offer modelling constructs by which *any* constraint could be put on a database. The author thinks that such ability is needed when creating consistent databases, where the description of the context of a geographical entity is an important factor affecting the integrity of a database.

This paper has represented some modelling constructs, and an outline for the definition of spatial relationships between the constructs. What still needs developing is the object modelling of hierarchies between area partitions, and how their integrity is enforced as methods of objects. The modelling of spatial relationships between complex geometrical object types also demands further studies.

ACKNOWLEDGEMENT
This study has been carried out in Finnish Geodetic Institute, in the Department of Cartography. The author would like to thank Prof. Tapani Sarjakoski, who has given motivation for this work. The discussions with him have greatly contributed to my understanding of the problem area.

7 REFERENCES

Booch, G. 1986. Object-oriented Development. IEEE Trans. Software Engineering, SE-12 (2) : 211-221.

Egenhofer, M.J. & Franzosa, R.D. 1991. Point-set Topological Spatial Relations. Int. J. Geographical Information Systems, 5 (2) : 161-174.

Elmasri, R. & Navathe, S. 1989. Fundamentals of Database Systems. The Benjamin/Cummings Publ. Co, Redwood City, California.

Kemppainen, H. & Sarjakoski, T. 1990. Object-oriented GIS - A Review and a Case Study. The Photogrammetric Journal of Finland, 12 (1) : 93-107.

Kim, W. 1991. Introduction to Object-oriented Databases. The MIT Press, Cambridge, Massachusetts.

Laurini, R. & Milleret-Raffort, F. 1991. Using Integrity Constraints for Checking Consistency of Spatial Databases. GIS/LIS '91, Atlanta, Georgia, pp. 634 -642.

Mark, D.M. & Frank, A.U. 1989. Concepts of Space and Spatial Language. AUTO CARTO 9. Ninth International Symposium on Computer-Assisted Cartography. Baltimore, Maryland, pp. 538-556.

Molenaar, M. 1991. Formal Data Structures, Object Dynamics and Consistency Rules. In: Ebner, H., Fritz, D. & Heipke, C. (Eds). Digital Photogrammetric Systems. Wichmann, Karlsruhe, pp. 262-273.

NCDCDS, 1987. Issues in Digital Cartographic Data Standards. Report #8, A Draft Proposed Standard for Digital Cartographic Data. National Committee for Digital Cartographic Data Standards, Columbus, Ohio.

Nyerges, T.L. 1991. Representing Geographical Meaning. In: Buttenfield, B.P. & McMaster, R.P. Map Generalization: Making Rules for Knowledge Representation. Longman Scientific & Technical, Bath Press, Avon, pp. 59-85.

Sowa, J.F. 1984. Conceptual structures. Information processing in mind and machine. Addison-Wesley Publ. Co.

Urban, S.D. & Delcambre, L.M.L. 1990. Constraint Analysis: A design Process for Specifying Operations on Objects. IEEE Trans. Knowledge and Data Engineering, 2 (4) : 391-400.

White, M.S. 1984. Technical Requirements and Standards for a Multipurpose Geographic Data System. The American Cartographer, 11 (1) : 15-26.