How to Design Geographic Databases? Specific UML Profile and Spatial OCL Applied to Wireless Ad Hoc Networks

Myoung-Ah Kang[1], François Pinet[2], Michel Schneider[1],
Jean-Pierre Chanet[2], Frédéric Vigier[2]
[1] Laboratory of Computer Science, Modelling and System Optimisation (LIMOS)
ISIMA / Clermont Ferrand University, France
{kang, schneider}@isima.fr
[2] Cemagref Clermont Ferrand, France
{francois.pinet, jean-pierre.chanet, frederic.vigier}@cemagref.fr

1. INTRODUCTION

This paper focuses on the description of a specific formalism for geographic database design and a complementary language for the expression of topological constraints. The proposed formalism enables designers to easily and clearly specify aspects related to geographic information. For that, our proposition uses concepts that are easy to understand for GIS (Geographic Information System) designers. Indeed, the main goal of our works is to respond to the needs of geographic database designers in offering a formalism more adapted to the field of GIS than classical modelling languages. Moreover, we also address the problem of precisely express topological constraints in complement of the geographic object description.

Firstly, this paper tries to define a GIS profile for the Unified Modelling Language (UML) (OMG, 2001; Page-Jones, 1999) and provides a data description particularly suitable for the modelling of geographic databases. This profile aims at specializing UML in order to clarify and facilitate the specification of spatial data. In order to reach this goal, the concept of geographic class is used; a geographic class has an attribute that can store geometries. This paper underlines that type constraints applied on class diagram are important to describe more precisely spatial features. These constraints are related to the spatial attribute of a geographic class in order to define more precisely what types of geometries are associated to the class (a point, a polygon, a set of polygons…). The constraints must be able to express the spatial composition (or aggregation), since it is an essential operation in geographic database (Tryfona, 1997). In this paper, a precise and easy to use textual representation of type constraints will be suggested.

Secondly, beyond the needs of modelling geographic objects, a part of this paper responds to designer requirements concerning the expression of topological constraints in class diagrams. In order to model these constraints, the existing propositions suggest to draw relationships between classes (Kösters, 1997; Parent, 1999); see the example in figure 1. This type of representation cannot express constraints depending on a specific condition (for example IF … THEN the constraint is applied ELSE …). Thus, in order to define topological constraints more precisely, an extension of OCL (Object Constraint Language) will be proposed and experimented. A final goal of this extension is to give the capability to generate automatically inside a GIS, a mechanism for database integrity checking. OCL is the new generation constraint language of UML (Schmid, 2002). This language has been developed by IBM and currently, the standard is maintained by the Object Management Group (OMG, 2001). A growing number of information system designers use OCL in complement of UML models and it becomes important to propose a unified framework for specifying spatial data and topological constraints with UML and OCL.

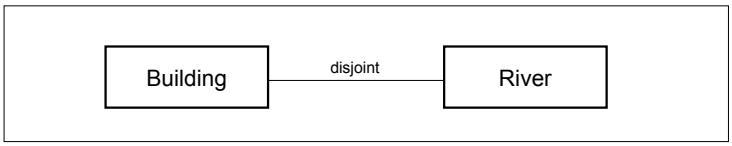

Figure 1: Two classes linked by a relationship "disjoint" in a class diagram; buildings and rivers are geographically disjoint.

Several examples will illustrate our proposition and at the end of the paper, we will present a complete case study that describes a database for wireless ad hoc networks. The design of this database type is currently under way in the context of a concrete project named RAHA. The next section of this paper introduces a UML extension presented in the form of a profile and dedicated to the geographic database modelling. The third section deals with an extension of OCL for expressing topological constraints between geographic objects. The fourth section presents a case study in order to show the usability of the proposed extensions. The last section concludes and draws some perspectives for future work.

2. TOWARDS A UNIFIED PROFILE FOR GEOGRAPHIC DATABASE MODELLING

2.1 Geographic class and type constraint

The concept of geographic class is introduced in order to clearly identify the geometry associated to an object. More precisely, each geographic class instance has a geographic feature. In practice, a geographic class includes an attribute having a spatial type and the value of this special attribute can store the geometry of a geographic entity. In this paper, this attribute will be called "geometry"; for example, a class Country can be viewed as a geographic class because each object of this class is associated to the geometry of a country. In order to describe real world entities, it is important that the geometry attribute can store data that are issued from spatial compositions; for example, a geometry of a Country class object can be composed by several simple polygons (continental parts and islands). This representation based on the use of a geometry attribute is especially suitable for spatial data design because it corresponds to data structures often used in geographic databases; for example, the GIS MapInfo supports this format (MapInfo Corporation, 2003). The implementation of a geographic class in a GIS is usually called a geographic layer. Constraints can be associated to geographic classes in order to indicate the types of geometry found in attribute values. Thanks to the constraint specification, in the example of the Country class, it becomes possible to express that each value of the geometry attribute corresponds to several simple polygons. In order to model spatial compositions, these constraints (called type constraints) are defined by textual expressions in which three types of operators can be combined:

- A XOR B; exclusive disjunction operator between A and B. For example "polygon XOR point" indicates that the value of the geometry attribute only contains either one polygon either one point (but not both).
- A AND B; conjunction operator between A and B. "polygon AND polyline" indicates that the value of the geometry attribute is only composed of one polygon and one polyline.
- A MULT *m*; multiplicity operator between A and a multiplicity *m*. "polygon MULT (1..5)" indicates that the value of the geometry attribute is only composed of "from 1 to 5 polygons".

Thus, the type constraint "polygon XOR (point AND polygon)" associated to a class C indicates that the geometry attribute of C must only contain one element having the type polygon or a composition of one element having the type point with one element having the type polygon; in other words, the value of the geometry attribute must contain only "one polygon" or only "one point and one polygon". Also, the type constraint "(point AND polygon) MULT (1..*)" expresses that the geometry attribute value must only contain point(s) and polygon(s); according to the constraint, the geometry attribute value must

contain the same number of points as the number of polygons. Note that a same type of operators can be found several times in a constraint expression. The combined use of these three operators types (XOR, AND, MULT) in textual constraint expressions gives the possibility of modelling all alternative and complex geometry types (in the sense defined by ISO and Open GIS Consortium standards); see (Brodeur, 2000) for information about spatial type standards in geomatics. It appears important to bring out an easy to use profile dedicated to geographic database design, and completely based on UML extension mechanisms such as stereotypes or tagged values (OMG, 2001; Page-Jones, 1999). Thus, the next subsection proposes a precise profile definition based on geographic class concept, type constraint definition and standard UML extensions.

2.2 Profile definition

Geographic Data Type. Geographic class instances are objects having a specific attribute named geometry. A class that specializes a geographic class is also geographic. A stereotype <<geographic>> is associated to each geographic class. The type of the geometry attribute is an unordered collection. This collection is more precisely a bag (also called multiset) and can contain the following simple geographic elements: point, polyline, simple polygon. This data structure based on bags corresponds to the perception of existing GIS data model by users; indeed, for GIS users, a total order on geographic objects is generally not needed and two objects can occupy the same XY location.

Type Constraint. In order to easily define combinations of geographic types, this paper proposes to use a non-ambiguous textual representation of type constraints. In fact, constraints can be set on the contents of the collections (i.e. on the contents of the bags). In using three "textual" operators (XOR, AND, MULT) and the geographic type names (point, polyline, polygon), it is possible to express what kind of elements can be contained in collections. In order to model a type constraint, a new tagged value geoTypeConstraint is introduced in geographic classes. For example, the tagged value {geoTypeConstraint=point XOR (polygon MULT (0..*))} declared in the definition of a class C, expresses that the geometry attribute value of C can only contain "one point" or "from 0 to *n* simple polygons". With this type constraint, the form of the corresponding geometry attribute value is $Bag\{\}$, $Bag\{point_1\}$ or $Bag\{polygon_1,\ldots, polygon_n\}$ i.e. an empty bag, a bag containing a point, or a bag containing from 1 to *n* polygons. The three operators XOR, AND, MULT are especially suitable in the case of geographical database design because they can express the spatial feature of objects issued from spatial composition operations; see (Pinet, 2001) for details. The next example illustrates the introduced concepts.

Example. We consider data illustrated in the figure 2. The map of a town is divided into school sectors. Each child living in a sector must be registered at the school corresponding to the sector and several sector divisions exist depending on the study level. Sectors of figure 2 consider for example two types of study levels (primary school sectors and secondary school sectors), and consequently two types of sector divisions. A sector is a polygon and a school is a set of polygons (i.e. a set of buildings) or a point depending on the available representation. Figure 3 describes a class diagram corresponding to the school sector example in using the proposed profile for geographic database design. Class Sector and class School are geographic; the geoTypeConstraint is used in classes in order to indicate the geometry of each entity. The type constraint appearing in the class School combines two operators (XOR, MULT).

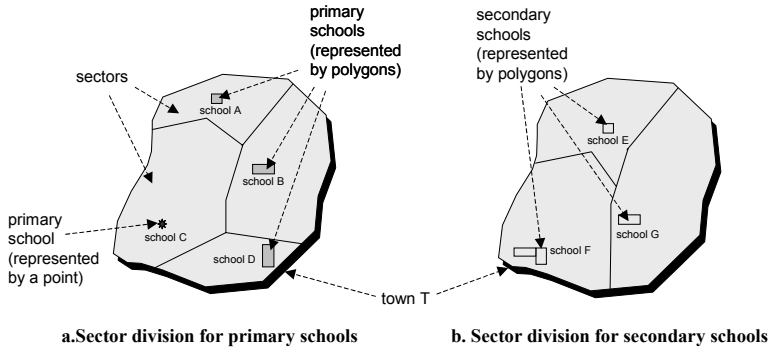

Figure 2: The school sectors example; two types of division for a same town.

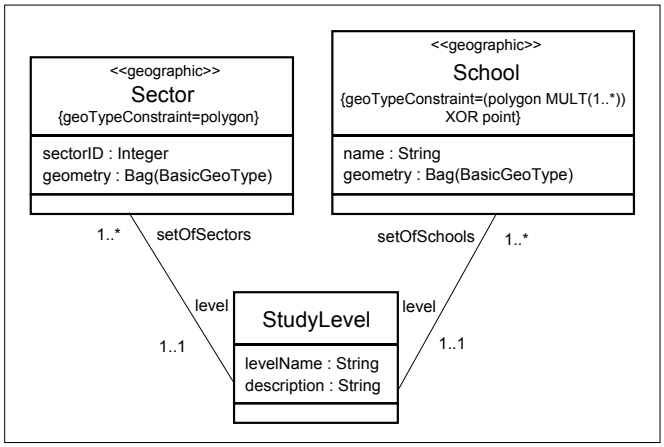

Figure 3: school sector model.

3. OCL EXTENSION FOR TOPOLOGICAL CONSTRAINTS

Topological constraints provide significant information on geometries. In the context of geographic databases, a specification of topological constraints underlines what types of spatial queries can be applied between classes. A spatial query is a database query implying a topological relation e.g. what is the name of the schools geographically located inside the sector 1; all GIS support this specific type of queries. Constraints can also be defined in order to specify consistency conditions in databases. Existing propositions allow the topological constraint expression by the use of relationships between classes (Kösters, 1997; Parent, 1999). For example, in diagram of figure 3, a relationship "contains…isInside" can be drawn between the class Sector and the class School in order to specify that each sector geographically contains schools. Nevertheless, in our case, this type of representation doesn't express complex constraints on geographic elements stored in a same value of the geometry attribute e.g. in the diagram of figure 3, it is not possible to model by adding relationships, that all buildings composing a same school are necessarily adjacent (in figure 2 the two buildings composing the school F are adjacent). Moreover, the representation by relationships cannot express constraints depending on a specific condition (for example IF … THEN the constraint is applied ELSE …). For these reasons, this paper doesn't explore the methods based on relationships but introduces a new OCL extension especially

adapted to the profile for geographic database design. OCL belongs to a new generation of constraint languages and provides a standard method for expressing invariants in UML class diagrams. The language is based on the easy to use concept of navigation between class instances. An overview of OCL can be found in (Schmid, 2002).

3.1 New OCL basic types

To extend OCL, new basic types are added, as presented in figure 4. The model of the figure extends a part of the OCL meta-model defined by (Richters, 1999). Note that instances of classes in this meta-model are the types themselves, not instances of the domain they represent. Thus, three new basic geographic types generalized by BasicGeoType are defined and each new type corresponds to a simple geographic type. In fact, each geometry attribute value (see section 2.2) is a bag of elements and each element in the bag has a BasicGeoType.

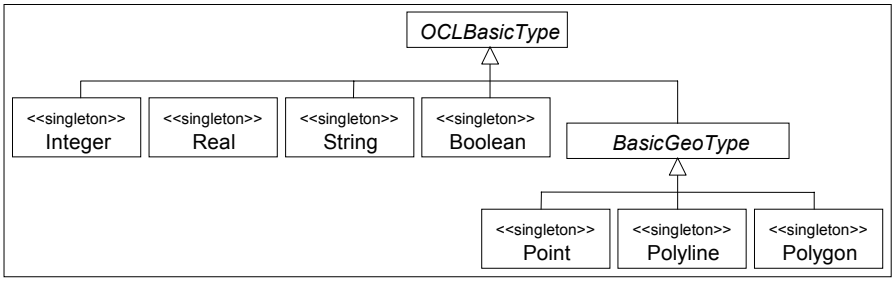

Figure 4: New OCL basic types (point, polyline and simple polygon).

3.2 Simple topological constraints between geometries

In order to describe topological constraints, specific OCL operations are defined. In this paper, spatial relations underlined by (ISO/IEC, 1994) have been chosen in order to illustrate our extension: overlaps, contains, is inside, are adjacent, are disjoint, are equal. Each of these relations are considered between a pair of simple geometries (point, polyline, polygon). The work presented in (Egenhofer, 1991) indicates how it is possible to check topological relations between two geometries having a different type (for example between a point and a polygon). The proposed OCL extension introduces six operations for checking the described relations; one operation for each topological relation. These operations can appear in OCL expressions and can be applied between objects having the type BasicGeoType or the type *Bag*(BasicGeoType) i.e. the type "bag of elements that have a BasicGeoType". These operations are described in table 1 and the generic algorithm used for checking a topological constraint is defined in figure 5. Objects having a geometry attribute value equal to null are simply ignored by the algorithm. Indeed, null values often correspond to undefined or unknown data, and it seems preferable that this specific information doesn't negatively influence the topological constraint checking.

Operations	Description
g1`->overlaps(`*g2*`)` *g1*`->contains(`*g2*`)` *g1*`->isInside(`*g2*`)` *g1*`->areAdjacent(`*g2*`)` *g1*`->areDisjoint(`*g2*`)` *g1*`->areEqual(`*g2*`)`	We define that *g1* and *g2* are the parameters of the operations. The type of *g1* and *g2* is BasicGeoType or *Bag*(BasicGeoType). These operations return true or false (a boolean) depending on whether the topological relation between *g1* and *g2* is true or false. The generic definition of the operations is described in figure 5.

Table 1: New OCL operations for checking topological constraints.

Our approach is generic and the GIS designer can easily choose other topologic relations such as Egenhofer relations for example (Egenhofer, 1991). In this case, only a new definition of the relations for couples of geometries must be established.

Concretely, the topologic operations presented in table 1 can have as parameters the geometry attribute (type Bag(BasicGeoType)) or in the case of a geometry composed by several elements (i.e. several parts), one element of the attribute geometry (type BasicGeoType). It is important to note that as presented in figure 5, if a parameter has the type Bag(BasicGeoType) and if its size is >1 then the operations return systematically false. Furthermore, elements of the geometry attribute can be accessed thanks to the OCL standard operations used on bags ("forAll", "exists" or "select"); indeed, these operations can be directly applied on geometry attributes e.g. "for each element e in the geometry attribute value" is written as "geometry->forAll(e|…)".

```
input: g1 and g2
output: true or false
   //The topological relation can only be checked between two simple
   //geometries. If g1 or g2 is a bag then it must contain only one element.
if             ( the type of g1 is Bag(BasicGeoType) and (size of g1)>1 ) or
        ( the type of g2 is Bag(BasicGeoType) and (size of g2)>1 )
then return false;
else {
   //The topological relation is always true if one collection is empty.
        if ( the type of g1 is Bag(BasicGeoType) and (size of g1)=0 ) or
                  ( the type of g2 is Bag(BasicGeoType) and (size of g2)=0 )
               then return true;
else {                  if the topological relation is true between
                      g1 and g2 then return true;
    else return false; } }
```

Figure 5: Generic algorithm used by operations for checking if a topological relation is true or false.

3.3 Practical examples

As illustrated in the following examples, the proposed OCL extension offsets the limitation of existing works notably concerning the expression of complex constraints depending on a specific condition (for example IF … THEN the constraint is applied ELSE …).

The following constraint refers to the school sector example described in figure 3. This constraint expresses that there is no school which is not inside a sector i.e. each part of a school is in a sector.

```
context School inv:
self.geometry->forAll(g|Sector.allInstances->exists
(sec|sec.geometry->contains(g)))
```

Let self be a school. The previous OCL expression defines that for each part (i.e. each building) of the self geometry, a sector sec containing self exists; this is a constraint between objects belonging to different classes. The following OCL expression corresponds to a constraint between instances of a same class. It describes that sectors associated to the same study level are disjoint or adjacent. For example, the four sectors of the primary study level described in figure 2 are adjacent or disjoint.

```
context Sector inv:
Sector.allInstances->forAll
  (sec|sec.StudyLevel=self.StudyLevel
 and sec<>self implies
  (sec.geometry->areAdjacent(self.geometry)
 or sec.geometry->areDisjoint(self.geometry)) )
```

As demonstrated in the next OCL expression, a constraint can be applied on a geographic type. This constraint means that secondary schools must be represented by polygons.

```
context School inv:
self.StudyLevel.levelName='secondary' implies
 self.geometry->forAll(g|g.oclIsKindof(Polygon))
```

Because the type of the geometry attribute is *Bag*(BasicGeoType), the variable g corresponds to one element of a geometry attribute value and consequently, the type of g is BasicGeoType. The basic type Polygon is defined in figure 4 and belongs to the extended OCL meta-model. The next constraint concerns an intra-object invariant i.e. constraints between geographic elements stored in a same value of the geometry attribute. The constraint models that all buildings composing a same school are adjacent. Indeed, for example, in figure 2, buildings composing the school F are adjacent.

```
context School inv:
self.geometry->forAll(g1,g2|g1<>g2  and g1.oclIsKindof(Polygon)
                                    and g2.oclIsKindof(Polygon)
implies g1->areAdjacent(g2))
```

3.4 Complex topological constraints between bags of geometries

Complex topological constraints between bags can be expressed thanks to the OCL operations "forAll", "exists" and "select". Let A and B be two bags. Figure 6 illustrates four complex constraints between A and B. Let CA and CB be two classes. We consider that CA has only one instance denoted by iA and CB has only one instance denoted by iB. The geometry attribute value of iA stores the bag A and the geometry attribute value of iB stores the bag B. Thanks to the proposed OCL extension, it is possible to write the four types of invariants described in figure 6:

Constraint a.

```
context C_A inv:
self.geometry->exists(g1|C_B.allInstances->forAll
       (i|i.geometry->exists(g2|g1->overlaps(g2))) )
```

Constraint b.

```
context C_A inv:
self.geometry->forAll(g1|C_B.allInstances->forAll
 (i|i.geometry->exists(g2|g1->overlaps(g2))) )
```

Constraint c.

```
context C_A inv:
self.geometry->exists(g1|C_B.allInstances->forAll
 (i|i.geometry->forAll(g2|g1->overlaps(g2))) )
```

Constraint d.

```
context C_A inv:
self.geometry->forAll(g1|C_B.allInstances->forAll
 (i|i.geometry->forAll(g2|g1->overlaps(g2))) )
```

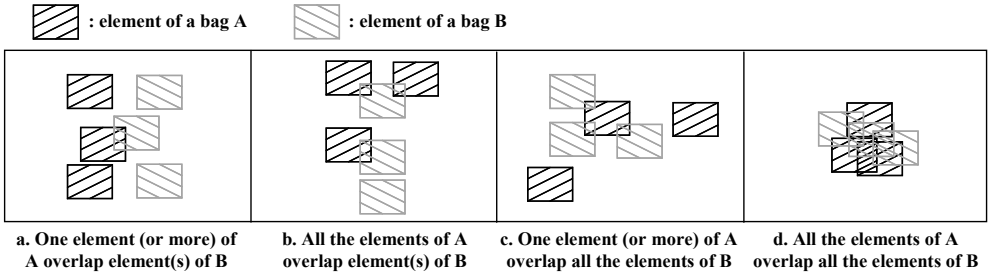

Figure 6: Topological constraints between two bags.

Thanks to OCL standard operations, other complex constraints can be defined e.g. constraints using conditions depending on the number of elements implied in a topological relation. The next example corresponds to the following constraint: only two elements of the bag A must overlap element(s) of the bag B.

```
context C_A inv:
self.geometry->select(g1|C_B.allInstances->forAll
(i|i.geometry->exists(g2|g1->overlaps(g2))))->size=2
```

4. CASE STUDY: MODELLING A WIRELESS AD HOC NETWORK DATABASE

This section deals with the description of a geographic database storing ad hoc network data. A Mobile Ad hoc NETwork (MANET) is a new generation of communicating systems. A node composing the network can communicate with another if these two nodes are in radio range i.e. if they are near. Some nodes can be mobile and in this case, a communication module can be embedded in a vehicle. Thus, the network topology constantly changes and messages must be transmitted from node to node. The first wireless node sends a message to a second wireless node which is a geographic neighbour. Then, the second node sends the message to another wireless node, etc. By this way, the message can cross the network. The case study of this section concerns the design of a database for monitoring each element of the network in a rural environment. The database is stored on a specific server and displays on a map the known elements of the system. In the model, a node knows its location thanks to the Global Positioning System (GPS); this information is transmitted to the server from node to node at regular time intervals. A wireless communication module has a retransmission range i.e. a geographical area in which the node can directly send a message without relay nodes; this area corresponds to the radio range. Figure 7 illustrates an instance of the database. The main server is located in a farm building and mobile nodes can be displayed on a road or in a shed.

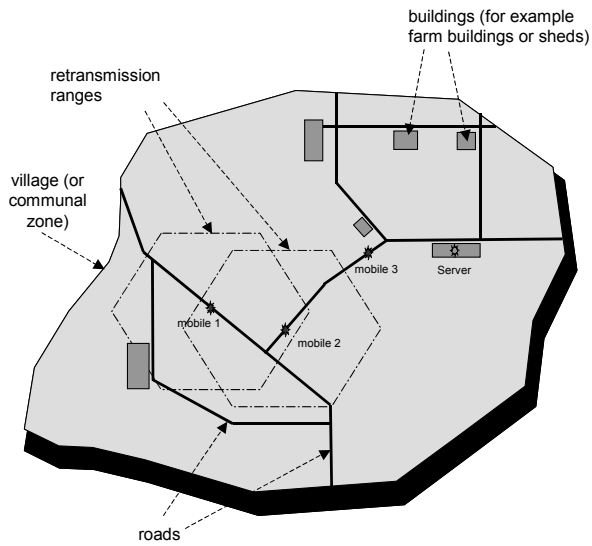

Figure 7: The ad hoc network example.

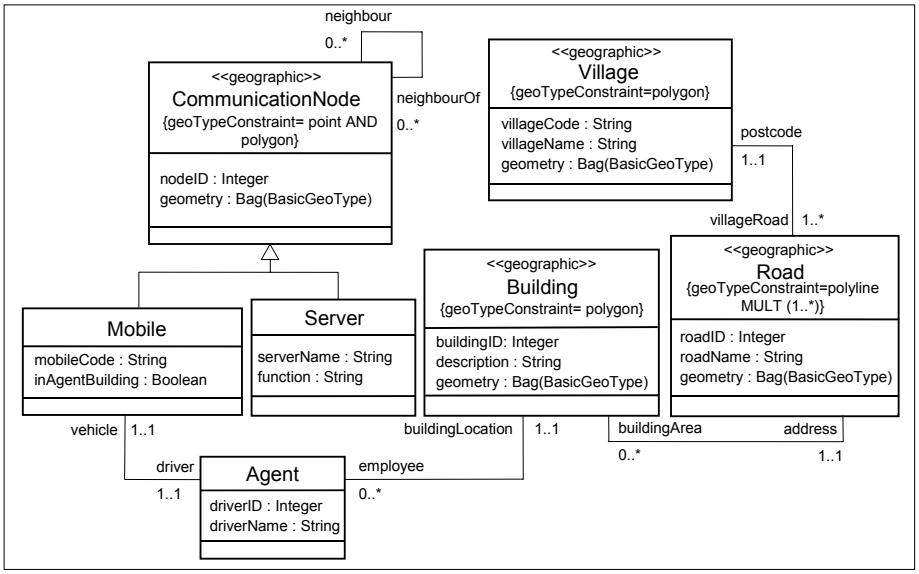

Figure 8: Ad hoc network modelling in using the unified profile for geographic database design.

Figure 8 presents the corresponding database conceptual schema. A communication node is geographically represented by a point (the node itself) and a polygon (the retransmission range associated to the node). The neighbours of a communication node are the nodes included inside its retransmission range. Agents drive an agricultural vehicle (a mobile node) and are employees in a shed. We give now the expression of some constraints which are useful for this database. For example, a server is installed in a building:

```
context Server inv:
self.geometry->forAll(g|g.oclIsKindof(Polygon) implies
 Building.allInstances->exists(bui|bui.geometry->contains(g)) )
```

Buildings and roads are adjacent or disjoint:

```
context Road inv:
self.geometry->forAll(g|Building.allInstances->forAll(bui|
bui.geometry->areAdjacent(g) or bui.geometry->areDisjoint(g)) )
```

All buildings are in their village:

```
context Building inv:
self.geometry->isInside(self.address.postcode.geometry)
```

If a first node is inside the retransmission range of a second node then the first node is the neighbour of the second one.

```
context CommunicationNode inv:
CommunicationNode.allInstances->forAll(nod1|
 nod1.geometry->forAll(g1|g1.oclIsKindof(Point) and
 self.geometry->forAll(g2|g2.oclIsKindof(Polygon) and
 nod1<>self and g1->isInside(g2) implies
  self.neighbour->exists(nod2|nod1=nod2) )))
```

If a vehicle enters the shed where the driver is employed, the attribute inAgentBuilding is true:

```
context Mobile inv:
  self.geometry->forAll(g| g.oclIsKindof(Point) and
   self.Agent.Building.geometry->contains(g) implies
    self.inAgentBuilding = true)
```

An interesting application of OCL expressions is the description of automatic database updates. Indeed, for example, the two last OCL constraints could define two automatic database updates. If the left part of the expression is true (the part before implies) then an update is made so that the right part becomes true.

5. CONCLUSIONS AND PERSPECTIVES

This paper describes a GIS profile based on the concepts of geographic class and type constraint. In complement to the profile, the proposed work provides a framework to express complex topological constraints thanks to a "spatial OCL". We have naturally chosen OCL because it is the standard constraint language associated to UML. In class diagrams, the existing formalisms only allow a topological constraint representation by relationships. By integrating topological operations to OCL, we propose a language for expressing spatial constraints depending on specific conditions. Also, as presented in section 3, the application of OCL operations like "forAll", "exists" or "select" on the geometry attribute allows very precise constraint expressions between bags of geometries.

Thus, our work aimed at extending UML models thanks to geographic type constraints and topological constraints. In using the proposed formalism, when the designer writes the system specifications, the visual representation of geographic data illustrated in figures 2 and 7 becomes unnecessary because all spatial features are declared in UML diagrams and in OCL expressions. The OCL extension can be easily adapted to new topological relations. In this case, the GIS designer simply has to give the semantics of the new topological relations between pairs of geometries.

The main goal of the proposed work is to automatically generate inside a GIS, a specific mechanism to check in real-time, the consistency of geographic data. In complement of the study presented in this paper, other research at the institute Cemagref is directed towards the generation of logical models thanks GIS profiles (Mirailles, 2003). Moreover, in the context of a specific project named RAHA implying LIMOS and Cemagref, an ad hoc wireless architecture is under way. The UML specification of this development will completely validate our profile and our OCL extension.

BIBLIOGRAPHY

Brodeur J., Bédard Y., Proulx M.J., Modelling Geospatial Application Databases using UML-based Repositories Aligned with International Standards in Geomatics. In Proc. of the Int. ACM Symposium on Advances in Geographic Information Systems. USA, 39-46, 2000.

Egenhofer M., Franzosa R., Point-Set Topological Spatial Relations. In Int. Journal of Geographical Information Systems: Vol.5(2). 161-174, 1991.

ISO/IEC SC21/WG3 N1680, ISO Working Draft SQL Multimedia and Application Packages (SQL/MM) – Part 3: Spatial. 150p, 1994.

Kösters G., Pagel B., Six H., GIS-Application Development with GeoOOA. In Int. Journal of Geographical Information Science: Vol.11(4). 307-335, 1997.

MapInfo Corporation, MapInfo Professional user's guide. USA, 2003.

Miralles A., GIS Profile for Objecteering, Cemagref, 2003.

OMG, Unified Modeling Language Version 1.4. OMG Report, 556p, 2001.

Page-Jones M., Fundamentals of Object-Oriented Design in UML. Addison-Wesley: 480p, 1999.

Parent C., Spaccapietra S., Zimanyi E., Spatio-Temporal Conceptual Models: Data Structures + Space + Time. In Proc. of the Int. ACM Symposium on Advances in Geographic Information Systems. USA, 26-33, 1999.

Pinet F., Lbath A., Semantics of Stereotype for Type Specification: Theory and Practice. In Proc. of the Int. Conference on Conceptual Modeling (ER'01). Springer Verlag: Japan, 339-353, 2001.
Richters M., Gogolla M., A Metamodel for OCL. In Proc. of the Conference on the Unified Modelling Language. USA, 156-171, 1999.
Schmid B., Warmer J., Clark T., Object Modeling With the OCL: The Rationale Behind the Object Constraint Language. Springer Verlag: 281p, 2002.
Tryfona N., Pfoser D., Hadzilacos T., Modeling Behavior of Geographic Objects: An Experience with the Object Modeling Technique. In Proc. of the Conference on Advanced Information Systems Engineering. Spain, 347-359, 1997.