GeoInformatica 4:1, 7–34 (2000)


# A Methodology for Spatial Consistency Improvement of Geographic Databases

SYLVIE SERVIGNE,[1] THIERRY UBEDA,[1] ALAIN PURICELLI,[1,2] AND ROBERT LAURINI[1]
[1]*Laboratoire d'Ingénierie des Systèmes d'Information (LISI), Institut National des Sciences Appliquées de Lyon (INSA), Bât 404, 20 avenue A. Einstein, 69621 Villeurbanne Cedex - France, E-mail: servigne, laurini@if.insa-lyon.fr*
[2]*Le Grand Lyon, Communauté Urbaine de Lyon, 20 Rue du Lac, BP 3103, 69399 Lyon Cedex 03*

Received July 17, 1997; Revised September 22, 1999; Accepted September 23, 1999

***Abstract***

In any information system the reliability of any results of queries, analysis or reasoning, depends on data quality (positional accuracy, consistency and so on). In some cases, answers cannot be obtained due to a lack of information, whereas in other cases answers are wrong or not complete because of inconsistent data. In geographical information systems (GIS), data quality management has to handle the spatial features of objects, which brings specific problems.

The goal of this paper is to describe a methodology for spatial consistency improvement of geographical data sets in vector format. It is based on errors survey and classification. Three kinds of errors are identified which lead to three kinds of consistency, namely structural consistency, geometric consistency and topo-semantic consistency. Each of them needs specific checking and correcting processes. All these processes are integrated in a general framework that is presented in this paper. An application of this framework to the Lyon Urban Community GIS (the SUR) is currently conducted; first results are presented.

**Keywords:** error checking, error correcting, spatial consistency, topological relation, data quality

## 1. Introduction

Data quality is a key issue of any information system. In GIS, geometric features of data make data quality management complex. Seven components of spatial data quality were defined [12] by the ICA Commission of Spatial Data Quality: lineage, positional accuracy, attribute accuracy, completeness, logical [10], semantic accuracy [16] and temporal information.

Our work concerns **logical consistency** and semantic accuracy aspects defined before. Regardless of the data sources (map digitizing, aerial photos, GPS data . . .), resulting geographical data sets must be consistent in order to be used in spatial analysis, and in order to ensure the reliability of important decisions based on geographical data (for example: concerning urban planning).

However, a lot of large existing geographical data sets suffer a lack of geometric and topological structuring resulting in errors. Therefore, the reliability of any results of queries, analysis or reasoning cannot be ensured (when a result can be obtained).

The goal of this paper is to present a general methodology **for spatial consistency**

**improvement of existing geographical data sets in vector format** in order to improve the results of data processing. Consistency between the database objects and the real world objects (loop1 on figure 1) as well as the internal consistency of the database itself (models and structures, loop2 on the figure 1) are both studied.

A general framework of consistency checking and correcting of geographical database in vector format is proposed.

In this paper, we introduce a new concept, namely the **topo-semantic consistency**, that is a subset of the **logical consistency** as it is defined by Kainz [10]. The topo-semantic consistency concerns the correctness of the topological relationship between two objects according to their semantic. For instance, a building inside another building is certainly an error whereas a building inside a parcel is not an error. And in both cases the relationship is a polygon inside a polygon. Therefore, the semantics attached to each object are mandatory to achieve the correctness of each relationship.

This paper is organized as followows: first, the spatial consistency context is presented and an consistency classification is proposed based on various kinds of possible errors (structural errors, geometric errors and topo-semantic errors). Then a consistency checking methodology is described based on topological and geometric characteristics of data. The methodology and correcting tools are combined into a general framework for checking and correcting of each kind of errors previously defined. The correction can be done manually by means of geometric tools or semi-automatically by means of correcting scenarios.

Before the conclusion, limitations of our detection and correction processes are sketched.

The methodology presented is actually conducted and experimented on the geographical database of the Urban Community of Lyon (city of Lyon plus some surrounding municipalities, that represents several millions of geographic objects in the database), called the ''Système Urbain de Références'', the SUR. This GIS has been implemented with the software APIC of APIC SA since 1985.

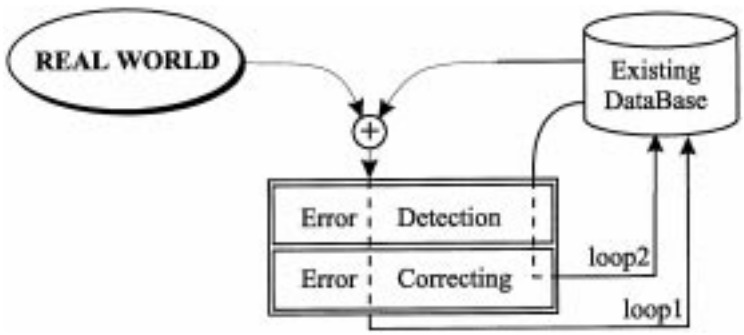


*Figure 1.* Consistency checking of existing geographical databases.

## 2. Spatial consistency context

Nowadays, GIS are more and more involved in processes in which reasoning is based on spatial features of objects. Cartography is no longer the main functionality of GIS.

Existing geographical data sets contain errors [11], and especially invisible errors (at working scale) that do not disturb the visualization, but that hinder or disable spatial reasoning. Furthermore, spatial data models of such GIS lack structuring to allow spatial reasoning. A consistency checking and correcting methodology is required to allow new spatial reasoning on geographic databases.

Our study deals only with **vector databases** and **existing geographic databases** without anyother information sources. To define a consistency checking and correcting methodology, only databases, data models and some data specifications are available.

Spatial consistency checking requires inconsistency detection processes as well as definition of **consistency errors**. An error can have different levels of consequences. Some can disable the reasoning and be so detected, whereas some can lead to erroneous results. In this last case it is difficult to know whether a result is reliable or not. This poses the problem of completeness in error inventory (are we sure to have identified all kind of errors?).

### *2.1. Spatial consistency errors and data sources*

The acquisition of data in a system can be carried out in different ways. The study of these different techniques gives hints concerning the origin of the errors. Usually, the different methods to obtain data are the following ones:

- *Capture of the coordinates directly on the screen* (with a pointing device): in such a case, the assessment is only visual and subjective (it entails an important inaccuracy); verifying tools that ensure a correct input are rare and moreover, they can sometimes be disabled.
- *Capture of coordinates directly with the keyboard*: problems of validity of data (where do they come from?, what is their degree of precision or quality?), data entry errors.
- *Automatic acquisition*: problems of data extraction like in aerial photos interpretation.
- *Transfer of data coming from other systems*: the degree of reliability and accuracy of data is not always known, errors can occur during the transfer. Semantic errors (based on topological relations for instance) due to the differences between these systems (various data structures, or systems of coordinates . . .) can appear.
- *Construction with other objects existing in the database*: propagation of the possible errors of the objects used.
- *Digitization*: precision of the positioning, pointing errors, multiple points (close to one another), objects captured twice.

It is then obvious that data acquisition processes can be sources of errors. Some of those errors can be found by a direct examination of data (a non-closed polygon, a point out of range) while some others require to take the semantics of objects into account (two

polygons that overlap can be an error or not, depending on the real world objects they are supposed to represent). Consequently, we have defined three kinds of errors, depending on which part of the objects we consider.

### *2.2. Kinds of spatial consistency errors*

The definition of each kind of errors depends on the level of the objects we have to look at to set its consistency. Three levels are defined:

- the structural level (data structures),
- the geometric level (conceptual model),
- the topo-semantic level (objects meaning according to topological relations).

These lead to three kinds of errors.

**Structural errors** come from data structures. Data structures must allow to store data according to the data model. Sometimes, specific characteristics handle by the data model are not handled by the data structures which can lead to errors. For example, in some GIS objects consisting in several polygons (some defining islands and others defining holes) are stored as *polygons with inner rings* (see figure 2). Inconsistencies can come from such cases of implementation. A data structure that doesn't faithfully implement a data model can lead to structural errors.

**Geometric errors** come from the geometric part of objects (the shape and the position). Data model must give a faithful representation of the world. Nevertheless, some features of the real objects geometry are sometimes not well captured by the data model. For example, a polygon must be closed, consequently a non-closed one is a geometric error (if the data model does not define polygon as closed object, the data model is then too weak).

**Topo-semantic errors** (topo for topological) are related to the meaning of the real objects represented in the database and to the topological relations they got with other object. Those errors consider and detect spatial relations.

## 3. Consistency checking

Geographic information modeling brings specific problems coming from the spatial attributes of objects. Real world objects such as buildings or lakes are characterized by a

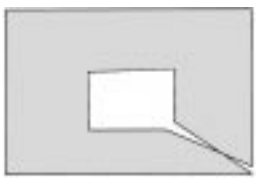

*Figure 2.* A bad polygon with inner rings.

form, a location, relations with other objects and semantics. The form and the location are called geometric attributes. The modeling process of geographical information must pay specific attention to these two kinds of attributes. Beside the geometric part of objects, spatial relations are very important. Spatial relations have been grouped into three categories [6]:

- topological relations which are invariant under topological transformations of the referenced objects,
- metric relations in terms of distances and directions,
- spatial order relations defining an order between objects depending on the observer.

Spatial consistency checking deals with all these features. The goal is to ensure that geometric and spatial attributes of geographical objects are correctly handled by the database. Figure 3 shows our methodology of spatial consistency checking. The starting points are the real world and the geographical database.

A list of properties has been extracted from observations of the world and from a study of the most used spatial data models. This list given on figure 4 is the subject of the next part (3.1). By picking up properties from this list according to the data model and quality specifications of the database, a suitable list for each GIS can be built. This final list of properties is then used to check the **geometric consistency** of the data set (see the central part of figure 3).

From the real world and mathematical theory, topological relations between objects can be described using the 9-intersection model, [3]. Topological integrity constraints can be defined using the topological relations. Constraints built this way are then used to check the **topo-semantic consistency** of the data set (see the left part of figure 3).

In a database, the data are stored using specific structures. These structures depend on the data model, but sometimes programming tricks have been used in order to handle specific attributes (for example the polygons with inner rings, see figure 2). This kind of error depends on the data structures and cannot be defined in a general way. Nevertheless, those problems must be taken into account and are defined as the **structural consistency** checking (see the right part of figure 3).

### *3.1. Geometric consistency*

In a database, the purpose of the data model is to give a representation of the real world. This representation must be simple and must capture the important features of real objects. Geometric modeling of objects must fulfill two basic needs:

- a good mathematical representation of objects,
- a good description of objects.

In existing GIS, data models answer these needs at different levels of completeness.

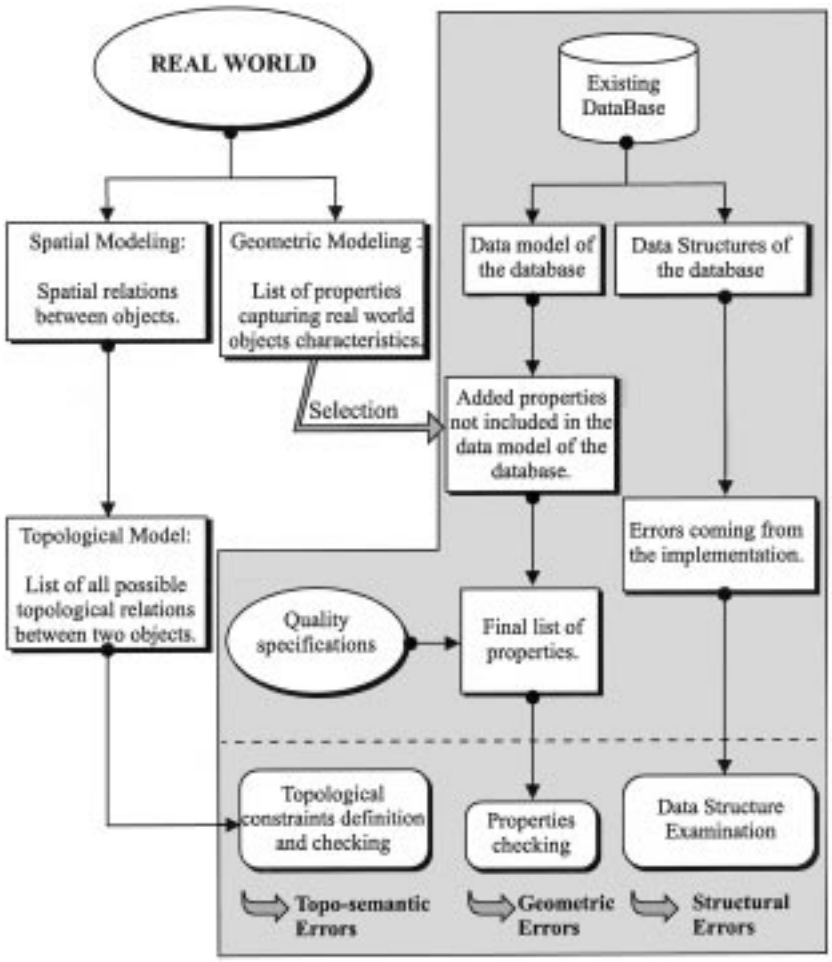


*Figure 3.* The consistency checking methodology.

Therefore, the first question to ask in a consistency checking process is: ''Does the data model capture all the relevant features of the real world objects?''

The answer is certainly ''no''. First of all, the real world contains too many semantics to be captured in a data model. Secondly, existing data models have been designed to answer needs (usually only cartography) that are no longer the only ones.

***3.1.1. The properties.*** Consequently, data models have to be improved by adding properties which capture spatial features of real objects.

A property is a rule on the shape or on the position of an object that must hold. Each time a property is not followed, a geometric error is detected.

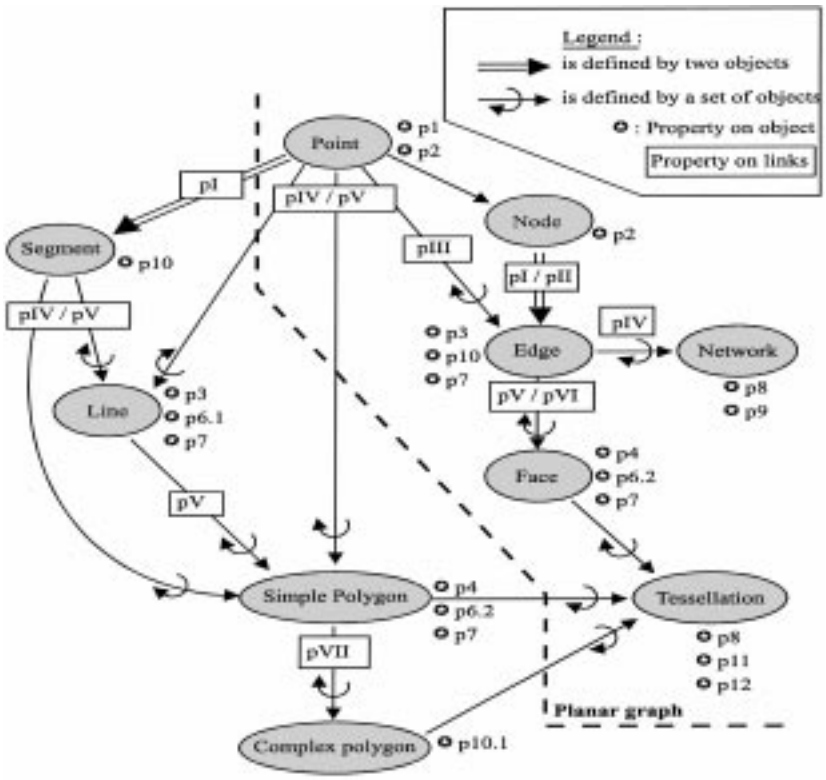


*Figure 4.* Properties on geometric objects and links between them.

Figure 4 gives a set of properties that can be applied to a database. Each property is attached to a shape or to a link between two shapes. The properties are the following:

**Properties on objects**

p1 Point consistency (range of value on each axis).
p2 Point or node uniqueness.
p3 Line or edge uniqueness.
p4 Polygon or face uniqueness.
p5 Referential existence.[1]
p6.1 Non closure of the object.
p6.2 Closure of the object.
p7 Non-self intersecting (of the boundary).
p8 Connectivity.
p9 Edge value.
p10 Orientation.

p10.1 Boundary orientation.
p11 Space coverage by a set of objects.
p12 Non overlapping polygons or faces.

**Properties on links between objects**
pI Two different points or nodes.
pII Belongs to at least two objects.
pIII Belongs to exactly one.
pIV All objects of the set are different.
pV Objects of the set are sorted.
pVI Belongs to exactly two objects.
pVII Belongs to at most two objects.

Depending on his data model, it is possible to build an appropriate list of properties suitable for a GIS by picking up properties from the list. This process is shown on figure 3 by the arrow called *selection*.

***3.1.2. The data model.*** Models based on spaghetti representation, planar graphs and polygonal data models are the most common geometric data models used in GIS (a description of these data models can be found in [18] and in [11]). They are based on geometric objects such as points (or nodes), lines (or arcs), polygons (or faces) and relations between those objects (a line is a set of points that can be sorted or not).

Data structures used to implement such models can contain some properties in their definition, i.e., stored objects automatically respect them. Consequently, properties captured by the data models need not to be checked. On figure 4, at the right of the dashed line, the list of properties to apply to a planar graph are given. This list can be found in [14].

### *3.2. Topo-semantic consistency*

In the methodology presented in this paper, only topological constraints are taken into account. A topological relation between two entities is based on the shared parts of their shapes. The validity of a given topological relation is based on the semantics (the meaning) of both entities. That is why errors coming from erroneous topological relations between entities are called topo-semantic errors.

Several models to handle topological relations have been designed [2], [3]. We present here the 9-intersection model and a method to design topological integrity constraints based on this model.

***3.2.1. Topological model.*** Topological integrity constraints are based on topological relation. Such relations describe the relative position of objects in the embedding space. We introduce here a model to handle those relations.

This topological model has been widely studied [1], [13] and has been given a rigorous

specification by Max J. Egenhofer, [3], [5]. In this model, binary topological relations between two objects A and B are defined in terms of the nine intersections of A's boundary ($\partial$A), A's interior ($A^\circ$) and A's exterior ($A^-$) with the boundary ($\partial$B), interior ($B^\circ$) and exterior ($B^-$) of B (see figure 5). Each object A and B can be a point, a line or a polygon. Definition of each part of each kind of geometric object is the following:

P is a point: $P = \partial P = P^\circ$.
L is a line: $\partial L$ = the two ending points of L.
$L^\circ = L - \partial L$.
R is a polygon: $\partial R$ = the intersection of the closure of R and the closure of the exterior of R.
$S^\circ$ = the union of all open sets in R.

For each intersection, the value empty ($\phi$) or non-empty ($\neg\phi$) is computed and stored into a $9 \times 9$ matrix:

$$\begin{pmatrix} \partial A \cap \partial B & \partial A \cap B^\circ & \partial A \cap B^- \\ A^\circ \cap \partial B & A^\circ \cap B^\circ & A^\circ \cap B^- \\ A^- \cap \partial B & A^- \cap B^\circ & A^- \cap B^- \end{pmatrix}$$

$\phi$ if the intersection is empty
$\neg\phi$ if the intersection is non-empty

*Figure 5.* The 9-intersection matrix.

***3.2.2. A model for topological integrity constraints definition.*** Topological integrity constraints (TIC) are defined using topological relations described by the 9-intersection model. The topological relation between two objects is the main part of the constraint. Considering the shape of objects, it is possible to compute all possible topological relations between two objects (according to the 9-intersection model). Considering the semantics of object (their meaning), it is possible to define which topological relation is consistent and which one is inconsistent.

A topological constraint is defined as the association of two geographical objects, a topological relation between them and a specification (see figure 6) which can be one of the following:

1. Forbidden
2. At least *n* times
3. At most *n* times
4. Exactly *n* times

The specification *forbidden* is the most interesting and usable one. Topological integrity constraints (TIC) defined using this specification are a mean for the end-users to describe topological situations they do not want to see in their database.

The 9-intersection model can be applied to all kinds of geometric objects. Considering


*Figure 6.* The definition of a topological constraint.

points, lines and polygons, it leads to six groups of relations: point/point, point/line, point/polygon, line/line, line/polygon, polygon/polygon.

This model allows to define only simple constraints (relation between 2 objects). In [8], T. Hadzilacos and N. Tryfona defined a model for topological integrity constraints also based on the Egenhofer model. Their model allows to define the same kind of constraints and to characterize complex spatial relations combining several single constraints. But it is less suited for end-users since the way to define constraints required the definition of logical sentences.

***3.2.3. An interface for topological integrity constraints definition.*** Topological integrity constraints define rules based on the semantics of the database entities. Like the list of properties, those constraints need to be adapted to the data set processed.

*Interface for topological constraints definition.* A visual interface to define topological integrity constraints was designed. Specifically, a dialogbox in which the user can choose a pair of entities, a topological relation, and a specification (see figure 7). Topological constraints are defined following the list of operations given here:

1. Choose a first class of entities.
2. Choose a second class of entities.
3. Choose a relation among the list proposed.
4. Define the specification.

In the case shown on figure 7, one of the constraints defined is:

```
(Road, Inside, Building, Forbidden)
```

The dialogbox shows a schema that illustrates the topological relation chosen in the constraint definition.

*Examples of TIC*

```
C1(Road, Cross, Building, Forbidden)
C2(Sluice, Joint, Waterpipe, Exactly 2 times)
```

*Note.* The 9-intersection model leads to 81 relations between points, lines and polygons [4]. This number is an impediment to the design of constraints for two reasons:

- there are too many relations to name each of them,
- to avoid a single situation to happening, one will have to create several constraints.

For these reasons topological relations have been grouped into subsets [18]. Using a subset in the definition of constraints will make the model more practicable for the user.

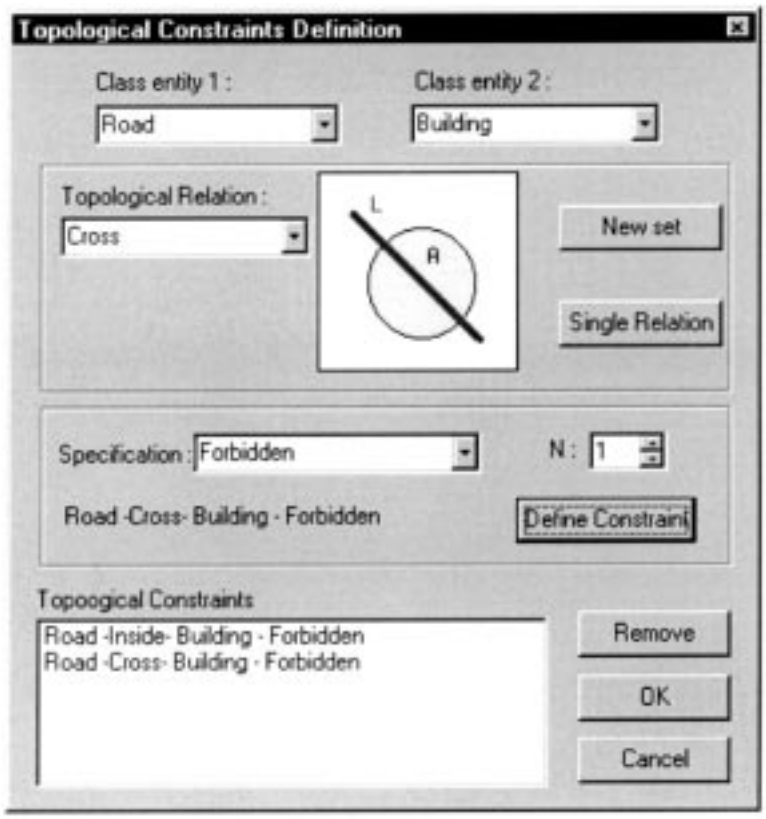


*Figure 7.* Topological integrity constraint definition interface.

***3.2.4. Predicates and functions.*** To compute topological relations we need to calculate the 9-intersection matrix (or at least some elements of it) that is to say the intersections of the boundaries, the interiors and the exteriors of the two objects. Each constraint can then be translated into a conjunction of verifications according to the relation (or subset of relation, see Section 3.2.2) involved in it.

Each subset or each single topological relation can be described by a partial relation. For example, a line crossing a line is defined by a single element of the 9-intersection matrix: $A^\circ \cap B^\circ = 0$, and leads to only one verification. Each verification is described by a predicate based on the dimension of the intersection ($\phi$, 0, 1 or 2, see first column of Table 1). In the previous example: INTERSECTION_DIM0($A^\circ$,$B^\circ$). In order to evaluate those predicates, functions to retrieve the boundary, the interior and the exterior of objects are required. The complete sentence for the previous example becomes: INTERSECTION_DIM0(interior(A),interior(B)).

For example, a one-dimension intersection between a polygon and a line, is a line. The predicate name is RL_share_L (Polygon and Line share a Line).

*Table 1*. Predicates definition.

| Type of Predicate | Object 1 | Object 2 | Predicate Name |
|---|---|---|---|
| INTERSECTION_DIM2 | polygon | polygon | RR_SHARE_R |
| INTERSECTION_DIM1 | polygon | line | RL_SHARE_L |
| | line | line | LL_SHARE_L |
| INTERSECTION_DIM0 | polygon | point | RP_SHARE_P |
| | line | line | LL_SHARE_P |
| | line | point | LP_SHARE_P |
| | point | point | PP_SHARE_P |
| INTERSECTION_EMPTY | polygon | polygon | RR_DISJOINT |
| | polygon | line | RL_DISJOINT |
| | polygon | point | RP_DISJOINT |
| | line | line | LL_DISJOINT |
| | line | point | LP_DISJOINT |
| | polygon | point | PP_DISJOINT |

Considering the four types of predicates and the three kinds of geometric objects, 9 functions and 13 predicates are enough to check all possible topological integrity constraints (see table 1 for the 13 predicates). The 9 functions are: boundary( point); interior( point); exterior( point); boundary(line); interior(line); exterior(line); boundary ( polygon); interior( polygon); exterior( polygon).

*Note*. Under the condition that we can calculate the boundary, the interior and the exterior of each kind of geometric object of the data set, it is possible to check the topological integrity constraints regardless of the data model used.

## 4. Framework for error detection and correcting

In this section, we present a general framework for data spatial consistency checking and correcting that will allow to apply the methodology introduced in the previous section to most of **vector data sets**. Some examples concerning practical experiences are also presented.

The framework on figure 8 presents links between different parts of a complete GIS, in order to check and to correct spatial errors. Two parts can be identified: the database itself which contains all information on the real world description and the GIS functionalities which contain a set of tools to access and to process the data set.

*The database*. We focus on the geometric representation and storage of the data in vector format. As in every database, a geographical database has a conceptual level and a

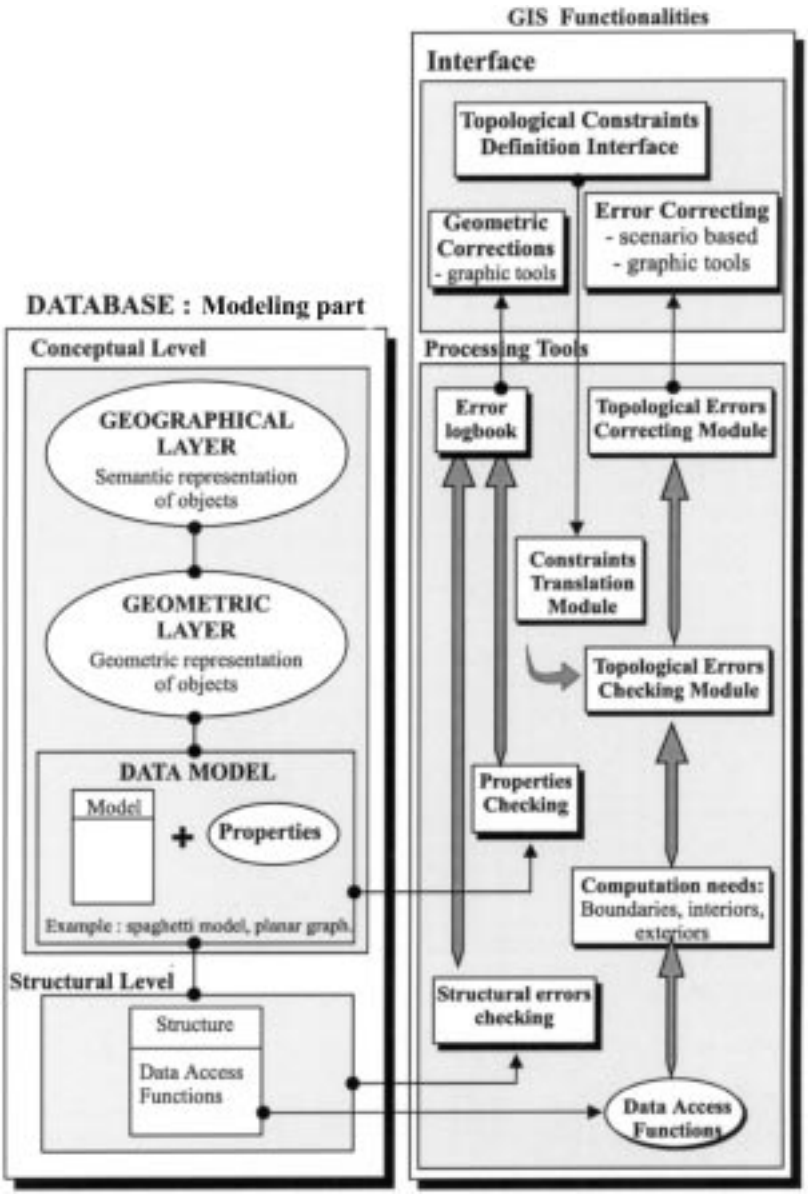


*Figure 8.* The framework for GIS consistency checking and correcting.

structural level. The conceptual level is the only one to be manipulated by the end-user. It presents the logical data organization in the database through the data model.

The structural level describes all the data structures used to store the geographical objects.

*The GIS functionalities.* It contains all the tools designed to manipulate the database. We distinguished two kinds of tools: the processing tools, that is to say processes that compute results of queries, analysis, reasoning and so on, and the interface tools which provide the end-user with means to define these queries. In our case, the interface contains a **topological constraints definition** interface, a **topological error correcting** interface which proposes all computed corrections for each error, and a **geometric error correcting** interface. The processing level contains data access functions (which depend on the data structures), error logbook management and a topological constraints checking process (constraints translation and checking).

In the following, both modeling and functional parts will be detailed.

First of all, we will define the adaptations needed to apply the methodology to a given database. The next sections will then describe the basis of the checking and the correcting processes applied to the different kinds of errors that have been introduced (structural, geometric and semantic).

### *4.1. Initialization: taking the database into account*

The first step of the methodology is to adapt the framework to the data set concerned.

Both semantic and geometric levels are subjected to specific adaptations.

***4.1.1. Selecting appropriate properties.*** The first stage is to extract from the general list given on figure 4, the properties which are relevant to the database in process.

First of all, we then have to refer to the geometric data model used in the GIS (the properties are obviously different depending on the model: spaghetti, planar graph . . .). We can then consult a possible document about quality (quality specifications), which gives indications about what needs to be verified or not.

We can thus expect to have an exhaustive list of the geometric properties (regarding to the data model and to the specifications) that will have to be respected in the database.

**Example:** The schema of the properties and data structures used in the SUR is presented on figure 9.

The software APIC only handles Points, Lines and Polygons (simple or not). All the properties on these entities (or between them) shown on figure 4 are relevant in this case.

***4.1.2. Defining topological integrity constraints.*** Using the visual interface presenting in figure 7, a list of topological constraints have been defined for the SUR.

**Example:** We present in Table 2 the set of topological integrity constraints we have defined to check the **cadastral layer** of the SUR (which is only a small subset of the entire database). The main entities contained in this layer are buildings, parcels, blocks and communes.

*Note.* The constraint in italic in table 2 is used to detect exceptions that are not real errors (building can be on several parcels). The aim of this constraint is only to list such cases. See Section 5.2 on exception handling for more details.

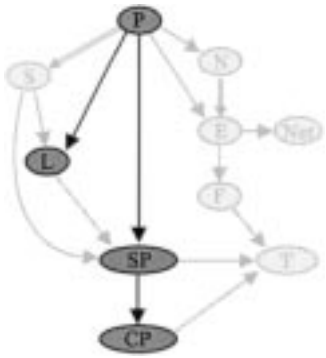


*Figure 9.* Properties of the SUR to be checked.

### *4.2. Detecting error*

Each kind of error is handled in a specific way. Consistency checking of structural errors (coming from programming choices) is handled by processes dedicated to a specific database.

Geometric errors are more general, they are linked to properties based on geometric objects commonly used in data models. Therefore, consistency checking processes designed for a given database can be re-used in another one. Nevertheless, some properties are very close to the data structures and the checking process attached to them need to be adapted.

Topo-Semantic errors depend on general concepts. Provided that specific data access functions have been implemented for the database, the checking processes presented can be applied to any GIS.

Generally speaking, checking processes meet with two difficulties: the problem of exhaustiveness of all kinds of errors likely to be met, and the problem of proving the completeness of each checking process:

*Table 2.* Topological integrity constraints of the cadastral layer.

| Entity 1 | Relation | Entity 2 | Specification |
|---|---|---|---|
| Commune | OVERLAP | Commune | Forbidden |
| | INSIDE | | Forbidden |
| Block | INSIDE | Commune | Exactly 1 time(s) |
| Block | OVERLAP | Block | Forbidden |
| | INSIDE | | Forbidden |
| Parcel | INSIDE | Block | Exactly 1 time(s) |
| Parcel | OVERLAP | Parcel | Forbidden |
| | INSIDE | | Forbidden |
| Building | INSIDE | Parcel | Exactly 1 time(s) |
| Building | OVERLAP | Building | Forbidden |
| | INSIDE | | Forbidden |

Are all the objects containing this error detected?
Are some good objects detected, (due to the limitations of the algorithm or to exceptions, that will be discussed in Section 5.3)?

***4.2.1. Structural errors.*** This kind of error cannot be established a priori. They only result from the data structures used in the database, so this list of errors cannot be determined at the beginning of the processing.

Those errors cannot always be checked by a property-verifying process. They often come from the programming tricks (for instance, the polygons with inner rings used to handle complex-polygons) used to handle cases that are not clearly defined by the data model. The number of tricks needed to handle the geometric representation increases with the weakness of the model. In contrast, a more complete model, with strong, constraining and clearly defined properties (such as in planar graph) reduces these structural errors.

The definition of general processes to check and to correct this kind of error is out of the scope of this paper, but is a prerequisite of the general consistency improvement process. Therefore, we present an example of a typical structural error found in the SUR and how we processed them, in order to illustrate this part.

*Polygons with inner rings.* Concerning the Lyon Urban Community, a few cases of structural errors can thus be listed in this category.

The most interesting one concerns polygons wih inner rings. This error comes from the fact that in APIC, simple and complex polygons have been merged into a common data structure. The holes are described as part of the boundary of the object, and are linked to the external lines by two transparent segments or lines (with opposite directions in order to ensure the closure of the polygon, see figure 10): they are called *connecting segments*.

Thus, two kinds of problems come from this particular means of representation:

a. The transparency of these connecting segments, and the inaccuracy of the coordinates can lead to erroneous polygons (self-intersecting boundaries, see figure 11a, or the hole is no longer recognized as one by the processes, see figure 11b).

   In the first case (11a) inside and outside boundaries have the same rotation direction (this error leads to a wrong result when computing the surface area of the object, for

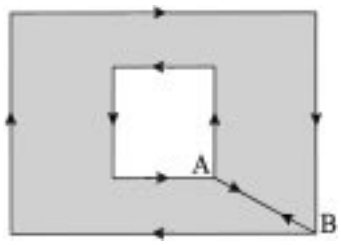


*Figure 10.* Polygon with inner rings.

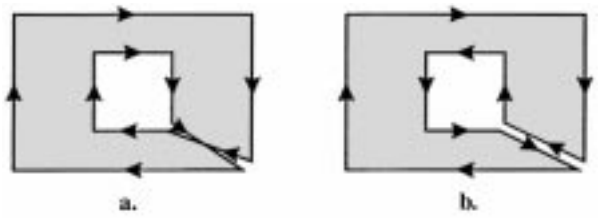


*Figure 11.* Connecting segments errors.

instance), whereas in the second one (11b), no property is transgressed (nevertheless it is really an error).

b. The second case of errors generated by these objects directly comes from the signification of these connecting segments. In fact, they do not take part in the description of the object; while numerous polygons come with invisible segments that really describe the objects, as shown in figure 12 (this was due to the cartographic tools of the software, in order to avoid drawing the same line several times). It is very important to make the distinction between these two kinds of transparent segments, so as to handle the connecting ones in a different way and not to take them into account when performing geometric processes ( perimeter calculus, buffer-zone . . .).

***4.2.2. Geometric errors.*** This kind of errors can be expressed as an object deficiency regarding to the properties of the model that we want to be respected.

Data checking depends on the data model. For most of properties a dedicated algorithm usable in most of data model can be defined (for example to ensure the closure of polygon, it is enough to have three algorithms depending on how the boundary is defined, using points, segments or arcs). Those verifications are a matter of computational geometry. A lot of useful algorithms can be found in [15], [21], [11], and see [9] for a presentation and a comparison of some algorithms: point-in-polygon, polygon-in-polygon, . . .Without a classification of vector data models in GIS, it is impossible to set up a complete list of checking algorithms, nevertheless a lot of simple cases can be handled the same way with well-known algorithms.

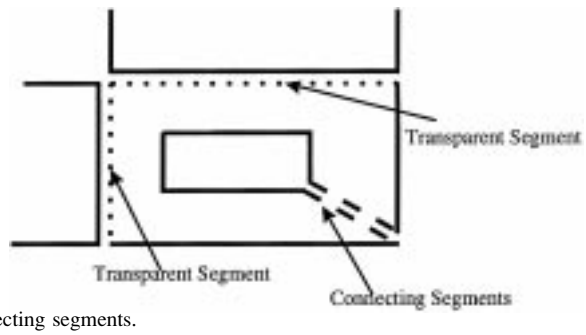


*Figure 12.* Connecting segments.

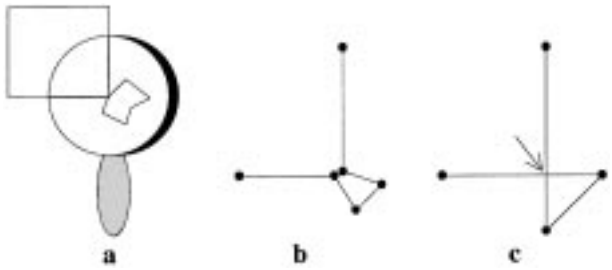


*Figure 13.* Weird polygons, superfluous points, forbidden intersection.

We present here an example of a kind of geometric errors found into the SUR and how we processed it.

*Weird polygons and useless points (figure 13a).* This kind of error comes from the digitizing process. Because of problems with the pointing device, several points have been acquired instead of a single one, resulting in weird polygon for example (figure 13c).

As shown on figure 13, this can result in two different kinds of situation:

The case on figure 13b can be detected by the uniqueness of point in a line or a polygon (property **pIV**: with a tolerance on the minimum distance between two points).

The case on figure 13c can be detected by the non self-intersecting polygon (property **p7**).

Furthermore, this case shows that property checking can sometimes be disturbed by implementation choice in the data structures used: within APIC, the curves are represented by a succession of points, close to one another so they visually look like curves (figure 14).

The difficulty is then to make the difference between points that define a curve (figure 14b) and real useless points (figure 14a).

Other cases of typical geometric errors are not presented here, such as duplicate points for instance.

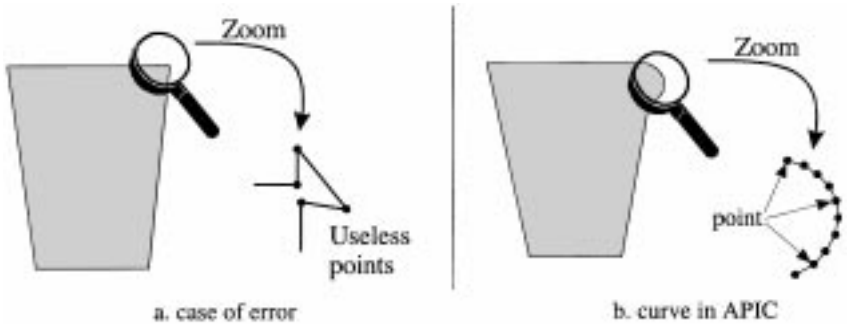


*Figure 14.* Weird polygon and definition of curves in APIC.

***4.2.3. Topo-semantic errors.*** Both the shape and the semantics of objects are required to check such errors. The shape allows to define the topological scene and the semantics allow to set the validity of the scene (by means of topological integrity constraints). The checking of topo-semantic errors requires topological relation computation. To simplify the problem, points, lines and polygons (or region) are the only shapes studied.

**For example**, the following topological integrity constraint has been defined:

(`River, cross, River, Forbidden`).

It is translated into:

**if** `INTERSECTION_DIM0(interior(River1), interior(River2))` **then** `Inconsistency detected.`

Each time an inconsistency is detected, the scene is stored into a logbook that will be used during the data correcting process.

***4.2.4. First results on the SUR.*** Algorithms have been implemented in the SUR in order to detect several kinds of errors (particularly those which have been presented). Thus, we have been able to estimate the number of objects to be corrected. The following results come from the application of these processes to a small and simple part of the SUR (the cadastral data of a single municipality). The Table 3, 4 and 5 present the data set size and the number of errors of each type that have been detected.

*Table 3*. Number of objects checked in the example data set.

| | |
|---|---|
| Municipality | 1 |
| Blocks | 142 |
| Parcels | 3352 |
| Buildings | 5466 |

*Table 4*. Number and percentage of errors of each type find in the data sets.

| | Polygons with Inner Rings | Double Points | Useless Points | Unconnected Points |
|---|---|---|---|---|
| Municipality | 0 | 0 | 1 | 0 |
| Blocks | 0 | 0 | 1 | 0 |
| Parcels | 10 (0.30%) | 2 (0.06%) | 57(1.70%) | 12(0.36%) |
| Buildings | 18 (0.33%) | 5 (0.09%) | 81(1.48%) | 1(0.02%) |

*Table 5*. Number of topo-semantic errors. Only very simple constraints have been defined in this example.

| | |
|---|---|
| Commune OVERLAP or INSIDE Municipality | no error |
| Block INSIDE Municipality | no error |
| Block OVERLAP or INSIDE Block | no error |
| Parcel INSIDE Block | 4 errors |
| Parcel OVERLAP or INSIDE Parcel | 10 overlaps |
| Buildings INSIDE Parcels | 118 errors (in fact, these are exceptions, see paragraph 5.3) |

### *4.3. Correcting errors*

In general cases, few automatic corrections are possible, because these corrections can be applied only if the detection process of the errors is complete, and if there is just one obvious possible correction. In the other cases, a manual correction is needed. It can be because the end-user has to verify the error is not an exception, or because the correction needs an interpretation (of the semantics of the object, for instance), that the computer cannot do. Then, the objective will be to propose to the user an interface in which different scenarios of corrections will be suggested, which will prevent him from introducing new errors (semi-automatic correction).

Concerning structural errors, the correcting processes will strongly depend on the database studied; in the other cases (geometric and semantic errors), some general ideas can be applied to any geographical databases. Nevertheless, the way of processing the corrections will necessarily depend on the data structures.

***4.3.1. Structural errors.*** As we have seen in Section 2.2, these errors directly come from choices of implementation. Thus, they will have to be corrected with regard to the data structures, and specific methods of correction will be defined for each case of structural errors.

Concerning the case of polygons with inner rings presented in chapter 4.2.1, we have decided to define a new style of line in order to distinguish the connecting segments from the transparent ones, since they do not have the same meaning, and to merge the ending points of those segments when necessary.

***4.3.2. Geometric errors.*** Most of the errors can be reduced to a problem concerning the points of the objects. Therefore, the different possibilities to handle points must be clearly defined; the following list presents basis operations that can apply to points:

- adding a new point,
- deletion of a point,
- merging two points,
- projecting a point on a segment,
- modifying the coordinates of an existing point.

**Example of correction: weird polygons with useless points**
The correcting process can be divided into three parts, as shown in figure 15:

- computation of the best location of the vertex
- projection
- deletion of the useless points.

Figure 15 shows only a very simple process that can be applied to this kind of error. This is the one we used into the SUR. Several ways can be used to find the best location for the

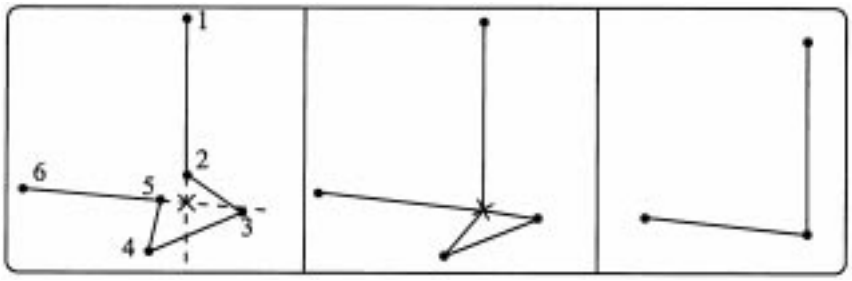


*Figure 15.* Weird polygon correction.

vertex: Center of gravity of the points, intersection of the two meaningful segments, use of other objects . . . (see [22] for more details on this topic).

***4.3.3. Topo-semantic errors.*** Only topological integrity constraints defined using the *forbidden* specification will lead to semi-automatic corrections. In this case, since an error is defined as a forbidden topological relation between two objects, the way to correct an error will be to change the topological relation between those objects. A set of correcting scenarios will be computed by applying several kinds of changes to both objects involved in the forbidden topological relation (together or one after the other). The changes proposed are the following:

- Objects modification:
  - Moving the objects.
  - Reshaping the objects.
- Deleting one object.
- Object splitting (creating an new object).

Computing and proposing correcting scenarios have two main advantages. The first one is to facilitate and to accelerate the end-user's work. The second one is to control the correcting process so that it can be ensured that the correction does not create a new error.

**Moving an object** ensures that the surface area of both objects remains unchanged. One of the two objects involved in the forbidden relation is moved according to a main direction until the topological relation changes. Applying this method to both ways of each main direction leads to a first set of correcting scenarios.

**Reshaping** means moving a part of an object and leaving the other part unchanged. The goal of such a correction is to change the topological relation between the two objects without changing the relations with the other objects of the data set. The adjustments will be made by some force-fitting algorithm that will snap characteristic points of one object onto characteristic points of the other object.

**Deleting one object** is useful when an object has been digitized twice. Two objects very close to each other can then be found. Two corrections are possible:

- keeping A and removing B
- keeping B and removing A

This leads to two scenarios.

*Note*. Removing an object is never easy in a database because of all the references to the removed objects that exist (e.g. spatial indexing). In this case, removing A and keeping B mean to replace A by B, and can be handled by a flag indicating to always use the object B instead of A in any reference. A complete removal of objects (and update of all the references) can be performed by a batch process afterwards.

**Splitting one object** into two new objects allows the maintenance of the planarity of a map. The only condition to check is that the two new topological relations are different from the previous one. Such a correction can be proposed when the forbidden relation is such as one of the two objects shares a part of its interior with the interior or the boundary of the other object. The corrections are:

- to split one of the two objects into several parts.
- to create a new object based on the shared part and remove it from each other object.

Figure 16 presents the correcting scenarios for two polygons sharing a part of their interiors. The relation between A and B is overlap.

The first possible correction is to remove the shared part from one of the objects (A of figure 16). The new topological relation is then Edge. This case covers two scenarios.

The second scenario creates a third object, named C that is the intersection of A and B, and subtracts it from both A and B. The new topological relations are:

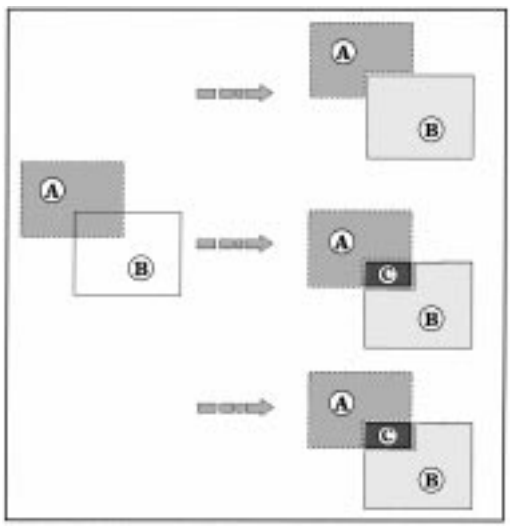


*Figure 16*. Polygon-polygon splitting scenarios.

- Disjoint between A and B.
- Edge between A and C.
- Edge between B and C.

The last one creates a third object named C, and subtracts it from only one of the two objects (A of figure 16). The new topological relations are:

- Edge between A and B
- Edge between A and C
- C Inside B.

This case covers two scenarios.

*Notes*. For each topological error, a list of correcting scenarios will be computed. The last step of the correcting process is then to choose which one to apply. To help the end-user to select the appropriate correction, the list of correcting scenarios will be presented using filtering and sorting processes. More details about each kind of change, scenario and about filtering and sorting process can be found in [19].

In the case of the SUR, the data structures used are based upon a spaghetti model. It sets up real problems in the correction of semantic errors, as they are based upon the topology of objects. In the worst case, the end-user will have to use the capturing and modification tools provided by the software. One has to remember that such a method can produce new errors in the database.

***4.3.4. Inter-layer problems.*** The examples already presented mostly concerned data belonging to the same layer. However, we can often have to handle several data layers, of which objects are not explicitly related (they are only related by their location). If an object was used as a model to create another object in another layer, the modification of the first one never affects the second one. This could then bring about new errors, that could be called ''inter-layers errors''. Such problems can be detected afterwards by the checking processes, but in that case, the corrections that have already been made on the first objects have to be repeated. In order to avoid these problems, processes enabling to copy these corrections will have to be found (for example with a detailed logbook of the corrections already made).

An interesting extension of the problem previously described concerns the systems the data of which could be updated externally (cadastral data for instance). In this case, as modifications have not been made in the system, they cannot have been copied on the other data layers of the system. Furthermore, the correction logbook might not be available. The problem is then to replace data of the other layers at the right place in relation to those which have been modified. Thus, a new step in this methodology will be to propose tools enabling to perform these operations automatically.

## 5. Limitations of the framework

### *5.1. Accuracy*

Checking processes often use a tolerance, which is in general a maximum distance between points, but can also be more geometric (angle of two segments). Any algorithm that uses a tolerance implies limitations: in fact, a tolerance must de defined as an absolute value: below this value, the object is correct (regarding the property checked), and beyond this value, it is incorrect. In practice, such a well-defined value cannot be found; therefore, a detecting process that uses a tolerance will never be complete: if the value of the tolerance is too small, errors can remain undetected, and on the contrary if it is too large, objects that are not errors can be detected. Thus, the tolerance will have to be set related to the objects to check, and to the precision of the data set, so that the detecting process will be as complete as possible.

### *5.2. Consistency of the correction*

The aim of such a methodology of correction is also to preserve this quality, during the complete process of course, and then after it. Several comments can be made:

First of all, the different correcting processes must not introduce new cases of errors (of any kind): this implies that these algorithms will have to ensure the accuracy of the corrected data regarding all the other kinds of errors. Thus these processes will only have to be performed once.

The second idea concerns the order of the corrections: as some errors can be corrected only if other problems have been solved before, the correcting processes will certainly be ordered: then we can suppose that we will correct the errors concerning just one object (structural and geometric errors) before those which concern several objects (semantic errors). A reverse method would come to try to do a sort of jigsaw with pieces that do not fit . . .

Lastly, when the database will be correct, the problem we will have to face will be to avoid that new errors appear in the database. Thus we will have to define processes that could check automatically every new object inserted in the database.

### *5.3. Exception handling*

Usually, integrity constraints define rules that data must follow. They are attached to a class of geographical objects and force each object of this class (road for example) to fulfill the same rules. It means that we expect the real world objects to have pre-defined characteristics.

Such an hypothesis is too strong. Exceptions always occur in geographical data sets. For example, most roads do not cross buildings, but some of them do. It does not mean that we cannot use such a constraint, but that we must provide a way to handle exceptions.

The topological integrity constraints we described previously in this paper allow to detect topological situations in order to correct them or just to report them. It means that the end-user does not have to correct each constraint violation detected in the database. He can also store them in a base of exceptions.

We propose two ways to handle exceptions. The first one is to define each constraint with a *no exceptions* attribute and to report each case as it is detected. The base of exceptions is then built little by little, by attaching a list of cases to each constraint. An exception is an expression containing a pair of objects: (`object1`, `object2`).

**Example:**

At the beginning,

`C1 = (Road, cross, Building, Forbidden) no exception.`

After the detection process,

`C1 = (Road, cross, Building, Forbidden) with exception (N7, B112); (N8,B114); etc.`

The second way is to allow the end-user to build up a first list of exceptions in advance. In this case, the pairs of objects are not checked during the detection process. In a hierarchical definition of geographical objects, we can define that integrity constraints are inherited through the hierarchy, and allow to define exceptions on subclasses.

**Example:**

Tunnel is a subclass of road.

`C1 = (Road, cross, Building, Forbidden) with exception (tunnel, building).`

## 6. Conclusion

In this paper, a framework for geographical data sets consistency checking and correcting was presented. It is based on three kinds of errors (structural, geometric and semantics) which have been defined in Section 2 and on a methodology of geometric consistency checking presented in Section 3. The framework is divided into three steps:

1 **Tailoring of the checking methodology.** According to the database, the set of properties to apply must be chosen and the set of topological integrity constraints must be defined.
2 **Properties and constraints checking.** The properties will be checked by means of computational geometry algorithms and the constraints will be checked by first order calculus predicates. This step results in a set of errors.
3 **Error correcting.** Scenarios of correction were computed for each. Semantic errors (coming from topological integrity constraints) and geometric tools are provided to handle specific cases.

In the correction processes, an essential aspect is to ensure that new errors are not created in the database. Automatic corrections clearly meet this need. As for manual corrections, they must propose different scenarios, that all ensure the consistency of the data.

This method designed to create topological integrity constraints is end-user oriented: compared to other approaches [8], it is not as complete, but easier to use. An objective will then be to integrate these other approaches in order to provide different levels of functioning, depending on the user's profile.

The methodology, as it has been described, has never been restricted to a specific data model (except for the examples of structural errors, of course). It forms an open system in which new sorts of errors can be inserted: we can never be certain of the completeness of the errors that can be found in the spatial database. Furthermore, new needs emerge that will reveal new limitations in the spatial data models (and new cases of errors).

All the examples presented in the paper come from the application of the framework to the Lyon Urban Community GIS. To apply the framework to another GIS, one will have to adapt it:

1. The structural consistency checking and correcting processes need to be re-designed according to the data structures of the GIS.
2. The list of properties for the GIS has to be set up. This is just a matter of selection.
3. The topological integrity constraints have to be defined for the GIS. The checking and correcting processes does not change.

An interesting perspective could be to design predefined sets of properties and processes according to data models or application types. First, specific properties sets can be set depending on most used data models in order to be used with the corresponding data model. Data models concern specific industrial models (ESRI, Intergraph . . .) but also generic models ( planar graphs, spaghetti, network . . .) depending on application types (regional planing, road management . . .). Another approach would be to allow constraints sets definition by area (in a single GIS) in which differences could be specified. The last proposition concerns constraints sets specification according to applications of the same type (e.g. the cadaster).

A further perspective concerns new data integration. Our methodology is applied to check consistency in existing databases. The aim is to use it to provide checking tools during data acquisition in order not to add wrong information and so, preserve the existing reliability.

## Note:

1. This property is not a spatial property. That is why it does not appear on the figure 4.

## References

1. J. Corbett. Cartographic data structures. Washington DC: US Census Bureau, 1979.
2. Z. Cui, A.G. Cohn, and D.A. Randell. ''Qualitative and Topological Relationships in Spatial Databases,'' in *Proceedings of the Third Symposium on Large Spatial Databases*, SSD'93, Singapore, June 23–25, Lecture Notes in Computer Science 692:296–315, 1993.
3. M.J. Egenhofer and J.R. Herring. ''A Mathematical Framework for the Definition of Topological Relationships,'' in *Proceedings of the 4th International Symposium on Spatial Data Handling*, SDH–90, Zurich, 803–813, 1990.

4. M.J. Egenhofer and J.R. Herring. Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases. Technical Report, Department of Surveying Engineering, University of Maine, Orono, 1991.
5. M.J. Egenhofer. ''Reasoning about Binary Topological Relations,'' in *Proceedings of the Second Symposium on Large Spatial Databases*, SSD'91, Zurich, Lecture Notes in Computer Science 525:143–159, 1991.
6. M.J. Egenhofer and R.D. Franzosa. *International Journal of Geographical Information Systems*, Vol. 5(2):161–174, 1991.
7. S. Faiz. Modélization, exploitation et visualization de l'information qualité dans les bases de données géographiques, Ph.D. thesis, University Paris Sud – Orsay, 1997.
8. T. Hadzilacos and N. Tryfona. ''A Model for Expressing Topological Integrity Constraints in Georaphic Databases,'' in *Proceedings of Theories and Methods of Spatio-Temporal Reasoning*, GIS–92, Pisa, 253–368, 1992.
9. D.P.W. Jayawardena and M.F. Worboys. ''The Role of Triangulation in Spatial Data Handling,'' *Innovation in GIS*, Vol. 2: Taylor & Francis, 7–17, 1995.
10. W. Kainz. ''Logical Consistency,'' in Elements of Spatial Data Quality, Elsevier Science, 109–138, 1995.
11. R. Laurini and D. Thompson. Fundamental of Spatial Information Systems. Academic Press, 1992.
12. J. Morrison, ''Spatial data quality,'' in Elements of spatial Data Quality, Elsevier Science, 1–12, 1995.
13. T. Peucker and N. Chrisman. ''Cartographic Data Structures,'' *The American Cartographer*, 55–69, 1975.
14. L. Plümer, ''Achieving Integrity and Topology in Geographical Information Systems,'' in *Proceedings of the First International Conference on Geographic Information Systems in Urban Regional and Environmental Planning*, Samos, GR, 45–60, 1996.
15. F.P. Preparata and M.L. Shamos. Computational Geometry: An Introduction, Springer Verlag New York, 1986.
16. F. Salge. ''Semantic accuracy,'' in *Elements of spatial Data Quality*, Elsevier Science, 139–152, 1995.
17. T. Ubeda and S. Servigne. ''Capturing Spatial Object Characteristics for Correcting and Reasoning,'' in *Proceedings of Joint European Conference and Exhibition on Geographical Information*, JEC–GI–96, Barcelona, Spain, 24–33, 1996.
18. T. Ubeda and S. Servigne. ''Geometric and Topological Consistency of Spatial Data,'' in *Proceedings of the First International Conference on Geocomputation*, Leeds, UK, 830–842, 1996.
19. T. Ubeda and M.J. Egenhofer. ''Topological Error Correcting in GIS,'' in *Proceedings of the fifth international symposium on spatial databases*, SSD–97, Berlin, Germany, 283–297, 1997.
20. H. Veregin and P. Hargitai. ''An evaluation matrix for geographical data quality,'' *Elements of spatial Data Quality*, Elsevier Science, 167–188, 1995.
21. M.F. Worboys. GIS A Computing Perspective, Taylor & Francis, 1995.
22. G. Zhang and J. Tulip. ''An Algorithm for the Avoidance of sliver Polygons and Cluster in Spatial Overlays,'' in *Proceedings of the fourth international Symposium on Spatial Data Handling*, SDH–90, Zurich, Switzerland, 141–150, 1991.

**Sylvie Servigne** received a Ph.D. in computer science in 1993. She is associate Professor since 1993 in the Computer Science Department and in the Laboratory of Information System Engineering both of the National

Institute of Applied Sciences of Lyon, France. Her research focuses on spatial information systems and more specially on data quality, continuous fields modeling and indexing, real-time GIS.

For more details: *http://www.insa-lyon.fr/People/LISI/Servigne/*

**Thierry Ubeda** received a Ph.D. in computer science in 1997. He is assistant Professor since 1998 in the Industrial Processing Department and in the Laboratory of Information System Engineering both of the National Institute of Applied Sciences of Lyon, France. His research focuses on spatial information systems and especially on data quality improvement, error checking and spatial modeling.

**Alain Puricelli** is a Ph.D. student of the Laboratory for Information Systems Engineering working in collaboration with the urban community of Lyon. His research focuses on spatial information systems and especially on data quality improvement, error checking and spatial modeling.

**Robert Laurini** is full professor, and director of the Laboratory for Information Systems Engineering shared by the National Institute for Applied Sciences (INSA) and the Claude Bernard University of Lyon. His main interests are spatial information systems, especially for urban and environmental planning. In 1992, with D. Thompson of the University of Maryland, he wrote ''Fundamentals of Spatial Information Systems'', Academic Press. He has authored or co-authored more than 100 papers. He is also teaching at the University of Venice (IUAV), Italy.