

## A service to customize the structure of a geographic dataset

Sandrine Balley<sup>1</sup>, Bénédicte Bucher<sup>1</sup>, Thérèse Libourel<sup>2</sup>

<sup>1</sup> Laboratoire COGIT - IGN, Saint Mandé, France  
{sandrine.balley, benedicte.bucher}@ign.fr

<sup>2</sup> LIRMM - CNRS - Université Montpellier 2, Montpellier, France  
libourel@lirmm.fr

**Abstract.** This paper deals with the usability of vector geographic data structures. We define usability of a geographic representation as the ability to fit the user's view, the user application and the user platform, plus the ability to be derived from a data producer dataset and to be maintained in the future if needed. Specifying the structure of a usable representation and deriving the corresponding data require specific tools and expertness. We propose to assist users in this process by means of a Web-based system able to assist users in specifying and performing a dataset restructuring process. The first step is to help users to set their requirements. To achieve this goal, we propose a graphical interface to specify differences between an existing data structure and a target data structure. The second step is to help users to commit these requirements into a transformation process applied to an existing dataset. We propose mechanisms to plan and execute this process and to check its result. The system relies on knowledge of existing data structures, platforms grammar rules and typical application schemas. The first part of this paper analyses the notion of a data structure and how to specify it. The second part describes our proposal.

### 1. Introduction

There is no best way to represent a portion of geographic space in a vector geographic database. Various users and applications, both of which are growing in number, have different best-fitted representations of geographic space. As they cannot design and create their own 'custom-made' dataset, users have to acquire them from an institutional data producer, but they wish to be provided with the amount and representation of information required by their application. For a national data provider such as IGN, enabling users to specify on the Web real world representations that are relevant to their application and providing them with the corresponding data is a great stake. By such a service users would keep benefiting from the data producer services to manage the dataset (quality checking and update), which is not the case if they adapt the data by themselves.

This paper introduces a service letting users specify their representation requirements starting from a representation proposed by the producer. These specifications

are processed by the system to generate a transformation chain to actually derive the data. We do not address the whole issue of deriving a best fitted representation but concentrate on the structural aspects of the representation. Moreover, we insist on setting the path for an easier maintenance of the user dataset.

The first part analyses the notion of a data structure and how to specify it. The second part presents our proposal to assist users in getting data with a usable structure.

## 2. How to formally describe a data structure ?

This part analyses what a data structure is and which languages are existing to describe it. Introduced concepts are summarized on figure 1.

The process of representing the real world in vector databases, i.e. abstracting real world entities into database objects, has been theoretically described [1] and practically documented [2]. The results of a representation process are the data, plus the framework to describe and encode this data. We called this framework the data structure. As shown in figure 1, designing a geographic data structure implies several steps: categorization, selection, modeling and implementation choices. These steps are successively defined below, together with existing solutions to document them.

*The categorization step* consists in defining categories of objects to be observed in the real world and somehow represented in the database, like 'road', 'forest', 'lake', 'building'. This step amounts to defining the ontology (or thesaurus) dedicated to the application domain of the representation designer.

*The selection step* consists in deciding which entities of these categories have to be represented in the database. A selection is usually expressed as filters, like 'roads that are dead-ends and that are shorter than 400 meters must not be represented in the database'.

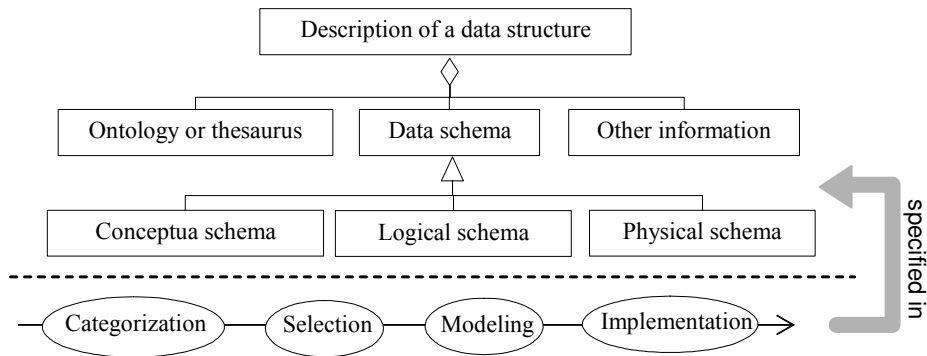
*The modeling step* consists in deciding how to represent selected real world entities as objects. For instance, roads might be represented by linear objects of the ROAD class, figuring their centerlines, with indications about their driving sense and administrative number. These objects must compose a network. The accessed amenities (such as tollgates and service stations) shall be explicitly attached through the PROVIDE ACCESS TO relationship. This step amounts to specifying the dataset conceptual schema plus some rules for data capture and consistency.

Rather advanced object types can be used in a conceptual schema (e.g. types with temporal primitives or multiple representations), depending on the used data model. For example the UML profile based on the OGC General Feature Model [3], the UML+PVL model used by the Perceptory CASE tool [4] and the MADS model [5] are extremely rich. As explained below, this is not the case for data schemas at a lower abstraction level, i.e. logical and physical schemas. That is why conceptual schemas cannot be derived from logical or physical ones.

Unfortunately, formalized conceptual data schemas are rarely provided to users. Defined by the data producer, they do not take part in the data transfer. It is notably the case for OGC WFS and WMS diffusion services.

*The implementation step* consists in specifying how the objects designed at the previous step can be represented in a machine-readable form. This step amounts to defining two data schemas. The first one is a platform-dependant logical schema. It describes how users actually manipulate data in the used platform, e.g. through tables and columns. The second one is a physical schema describing how data are stored. For example, it might be decided that roads are represented in a single file and that road numbers are encoded by strings up to 10 characters.

Unlike in conceptual schemas, only the object types that are allowed by the used platform grammar rules can appear in logical and physical schemas. An example grammar rule of the PostGIS DBMS is that topological primitives are not supported. Grammar rules have no standalone description, even in platform documentations. They are implicitly enclosed in data translators (transforming a logical schema into another) and in CASE tools (deriving a logical schema from a conceptual schema).



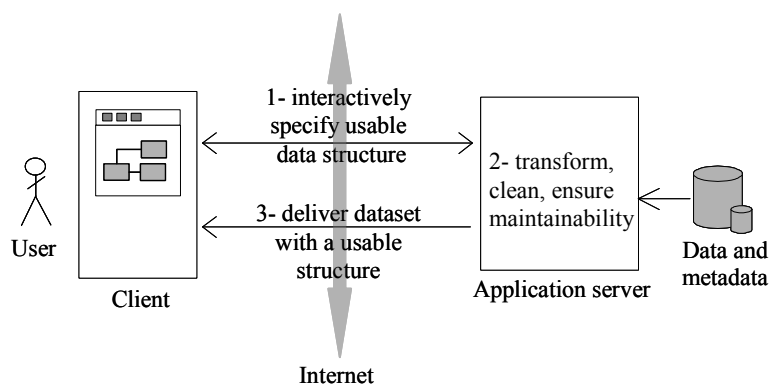
**Fig. 1.** Design steps to produce a structure to represent a portion of geographic space are displayed in the lower part. The upper part represents elements in which this design can be described.

The design steps of a data structure and their possible documentation are summarized in figure 1. It can be noticed that the entirety of a data structure can not be expressed by means of ontologies and data schemas. They appear as ‘other information’ in the figure. This is the case for the selection rules of real world entities, the capture conditions and the inner consistency rules defined at the modeling step. Most of the time this information is expressed through natural language statements in data specification documents. Different models tend to represent it more formally: among others, [1] proposes relations between an ontology of real world categories and a data schema. [6] proposes a spatial extension of the OCL formal language. [7] proposes a specification model including consistency constraints as an aspect of data quality.

This part has analyzed the notion of a data structure. It is a complex notion whose facets are unequally and independently described. As a consequence, it is difficult for a user to understand or to specify it at a glance. Our approach to assist users in specifying a usable data structure and deriving usable data consequently is presented below.

### 3. Helping users to restructure existing datasets into datasets with a usable structure

This part describes our proposal to assist users in deriving datasets with a usable structure in their context. The first difficulty experienced by users in this process is to identify their requirements concerning the structure of these datasets. This is addressed in section 3.1. The second difficulty is the very transformation of existing datasets into usable datasets meeting the requirements. This is addressed in section 3.2.



**Fig. 2.** Proposed data restructuring process.

We propose to handle these difficulties by means of a service helping users to get on-demand datasets. As shown on figure 2, the user first specifies its requirements on a graphical interface driven by an existing data structure. Relying on these requirements and some internal knowledge of constraints relative to data and data structures, the system automatically generates a suitable transformation chain. It last delivers usable data to the user.

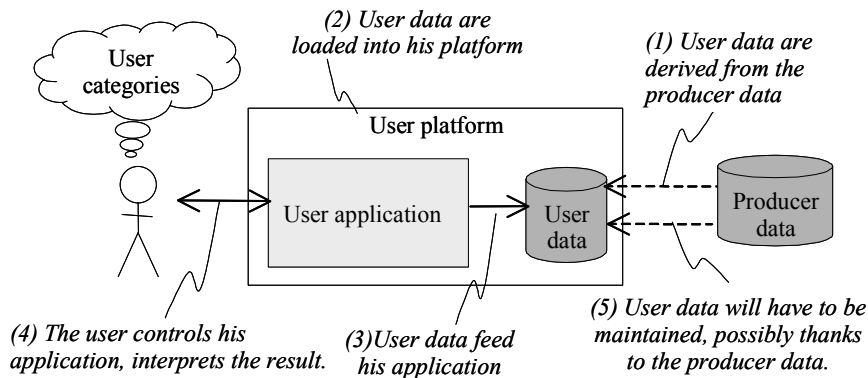
This approach may be compared to existing tools for modeling (CASE tools), translating (e.g. the Safe Software FME translator), restructuring (e.g. the Safe Software FME workbench), or checking (e.g. the Laser Scan Radius studio) geographic data. Some of these functionalities are also emerging on the Web [8]. These tools are specialized and powerful, be it to represent platform constraints or to transform data. However they do not provide any knowledge on data structures constraints, esp. at the conceptual level, which is the main characteristic of our approach.

### 3.1 What is a usable data structure ?

To define a usable data structure, we greatly rely on [9] description of geographical data usability: usability is considered as a ternary relationship between users and their needs, data and their characteristics, applications and their requirements.

We add two aspects to this definition in order to take into account the necessary relationship between the user and the data producer. This is represented on figure 3. In our work, a usable representation is:

- a representation that can be derived from an existing representation, for instance from an IGN data product (first requirement on figure 3),
- a representation that fulfills users requirements (second, third and fourth requirements on figure 3).
- a representation that can be maintained (fifth requirement of figure 3).



**Fig. 3.** Usable data representation must fulfill requirements of several processes.

### 3.2 Assisting users in specifying their requirements for a usable data structure

In our approach, requirements for a usable data structure are not expressed as a mere structure definition. As detailed hereafter, they are listed after to the five aspects considered in our definition of data usability (figure 3).

Firstly, a usable dataset must be derivable from some existing data. We propose to let users specify their structure requirements by stating differences between the needed structure and existing structures of data products diffused by a data producer like IGN. In our proposal, a two frame graphical interface (figure 5) provides a view of an existing data structure at the conceptual level, thanks to an object-oriented class diagram. This structure is stored in the system as a metadata. A user-defined data structure is designed by selecting and manipulating elements from the initial one. Design tools are used that provide functionalities such as renaming, splitting a class, dropping an attribute, etc.

Secondly, in a usable dataset, real world categories that are meaningful to the user must figure in the data structure. Depending on the producer dataset structure, they may not be represented by explicit classes in the conceptual schema. For example, in some IGN database, the 'bridge' category does not take part to the conceptual schema. To fulfill this requirement, we propose to enrich conceptual schemas with derivable implicit object types figuring a-priori meaningful categories. For example, the service provider could enrich the description of its conceptual schema with the implicit BRIDGE object type that can be derived from LINEAR CONSTRUCT and ROAD SECTION by means of predefined restructuration operations. Users can also define derived classes: by reading the structure description, they must assess if the required information is actually enclosed in the dataset. Then they must match implicit and existing object types by means of the design tools described previously.

Thirdly, a usable dataset must fulfill the requirements of the user application. Requirements on the inputs and outputs of an application are called pre-conditions and post-conditions. Here are listed the most frequent pre-conditions:

- Definition of the application schema, i.e. the conceptual data schema the most suitable for the application. For example, the application schema of any routing application is composed of ROAD and NODE object types explicitly linked by a topologic relationship.
- Platform constraints affecting the physical or logical schema: e.g. 'the service only reads GML data files',
- Information about setting input parameters affecting the process: e.g. 'setting parameters (e.g. a threshold, a tolerance value, etc.) can be set to zero for short itineraries', 'the used algorithm for route calculation may provide bad results if applied to very sinuous features'.

All of these pre-conditions cannot appear in application schemas. They are being further formalized to promote the discovery and chaining of data processing, especially those deployed as Web services [10] [11]. For our part, we only want to provide some general application schema templates (e.g. for route calculation or map making) to help users express the requirements issued by their application.

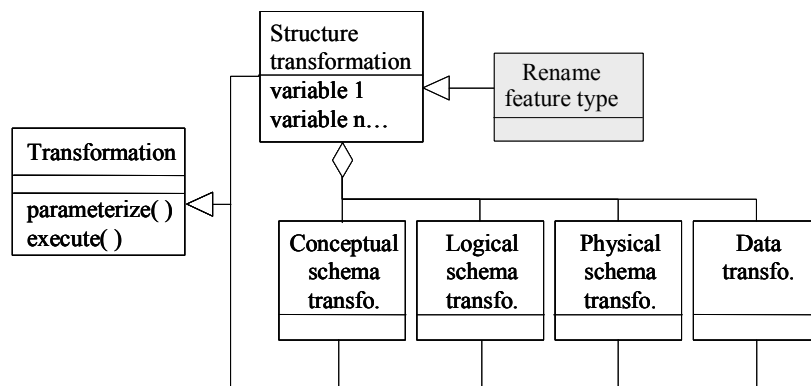
Fourthly, a usable dataset must fit the user platform, i.e. it must respect its grammar rules. Just as current CASE tools, we propose to prevent users from dealing with this technical issue. A first assistance we provide consists in generating suitable logical data schemas thanks to grammar rules of standard platforms published in our application. A second assistance consists in propagating conceptual schema evolutions to the logical level thanks to the stored links between these schema elements.

Fifthly, a usable dataset must be maintainable. To fulfill this requirement, we propose to trace transformations and to maintain correspondence links between initial objects, that are kept by the data producer, and the transformed ones. It enables to replay the restructuration process, to trace data errors and to propagate potential updates from the initial dataset to the user one.

In this section we have presented the main lines of our proposal to assist users in specifying requirements for a usable data structure. Users only express ‘applicative’ requirements. They are provided with existing conceptual schemas and some typical usable application schemas. Requirements on the logical and physical facets of data structures are automatically inferred by the system.

### 3.3 Assisting users in performing the transformation process

This section describes how our system assists users in performing the transformation process to derive a dataset compliant with the requirements specified above.



**Fig. 4.** The internal transformation model and an example of structure transformation (shaded class).

#### Applying schema transformations

As seen in the section before, users apply design tools that manipulate elements from an existing conceptual schema to specify their required structure. Our internal transformation model is shown on figure 4. Each design tool, e.g. ‘rename feature type’, is associated to a class extending the ‘Structure transformation’ class. A ‘structure transformation’ is an elementary step of a transformation process. It affects the whole data structure. It is composed of a ‘conceptual transformation’ (renaming a class of the conceptual schema), a ‘logical transformation’ (renaming a table), a ‘physical transformation’ (renaming a file) and a ‘data transformation’ (not affecting data in our example). Any ‘Transformation’ class has specific variables (e.g. ‘oldName’ and ‘newName’ for ‘rename feature type’). It also has two methods : a parameterization method to allocate values to variables, and an execution method to execute the transformation.

Applying a design tool automatically triggers the creation of an instance of the underlying structure transformation class. The system automatically invokes the parameterization method of this structure transformation. This applies and propagates variables assessment from the conceptual level to the lower levels. This propagation relies

on stored links between schema elements, and on an internal knowledge of used platform grammar rules. The system then invokes the execution method of the ‘conceptual transformation’ to transform the conceptual schema visualised by the user. The other transformation instances are inserted into a transformation plan for a later execution.

### **Planning and tracing the restructuration process**

When the user commits all his requirements and asks for the compliant data derivation, the system goes through its transformation plan and invokes the execution methods of successive transformations. Correspondences between the initial and transformed objects identifiers are stored.

During this transformation process, the system launches some additional expert mechanisms. As a matter of fact, the transformation process may lead to violate some general integrity rules (e.g., the existence of an association is conditioned by the existence of its member classes) or some specific ones stated in the data structure description (e.g., the road network is sectioned according to changes in the road attributes values). They trigger the parameterization of other transformations (e.g., some ROAD SECTION instances must be aggregated after the deletion of an attribute whose value was causing their separation).

## **3.4 Implementation**

A prototype of our application has been implemented within the GeOxygene open Java framework<sup>1</sup>. In GeOxygene, spatial data classes extending the OGC concept of Feature are mapped to a relational database via an object-relational bridge. In the current prototype, GeOxygene is also considered as the default user platform.

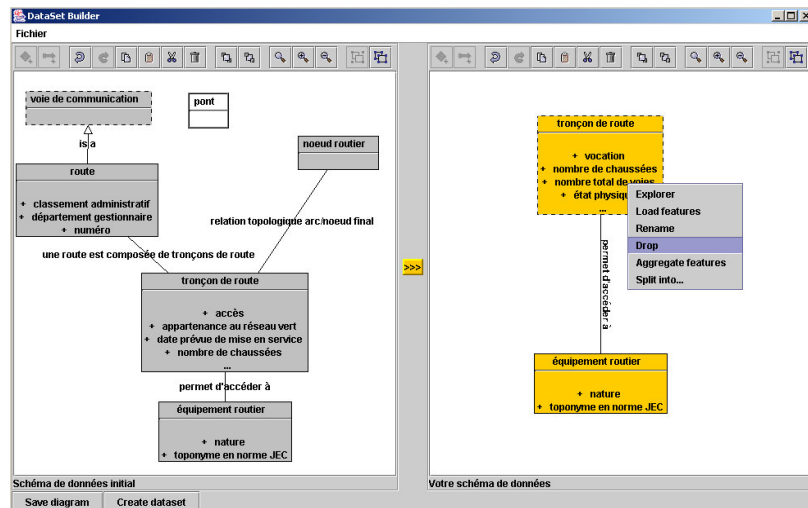
The description we propose for data structures is based on:

- the ISO 19109 General Feature Model, based on Feature Types, for the conceptual or application schemas,
- the “Class” and “Field” classes for describing the logical data schema in the Java platform
- the “Table”, “Column” and “Key” classes for describing the logical schema in the relational storage platform

The user interacts with the system through a client implemented as a Java Web Start application. This client displays a conceptual schema in a UML-like form based on the JGraph library. It interacts with our server through Web Services.

---

<sup>1</sup> <http://oxygene-project.sourceforge.net/>



**Fig. 5.** The user interface: the user data structure (on the right) is specified based on elements selected from an existing data structure (on the left). Design tools are proposed in a contextual menu. An implicit object type is displayed as a transparent class on the left.

### System evaluation

No user test has been carried out so far. An evaluation session is planned within the COGIT lab with a pre-defined use-case : the restructuration of a dataset for a map-making purpose. Of course, the COGIT users are more comfortable with geographic data structures than the users targeted by our system. A real condition evaluation is necessary to assess the usability of our system.

### Conclusion and perspectives

In this paper, the usability of a geographic representation is defined as its ability to suit the user's view, to fit into the user platform and application, to be derivable from an existing dataset and to be maintainable.

We propose to assist users in acquiring geographic data with a usable structure on the Web. Our approach aims to spare users from technical considerations and to let them focus on the application level. Starting from an existing data structure, a target data structure is interactively specified at the conceptual level. A complete transformation plan, including logical schema restructuration and data cleaning operations, is automatically designed by the system. This plan is executed on the initial data before they are sent to the user. The restructuration plan is stored, as well as links between initial and restructured datasets.

The main drawback of our system is that it requires formal descriptions of data structures, including an explicit mapping between schema elements at different abstraction levels. This issue is easily tackled in our case as the GeOxygene platform manages a data dictionary, a Java API and related tables in the same time thanks to

mapping files. However, to reuse our approach on datasets that are not stored in Ge-Oxygene, a solution must be found to acquire and manage such extensive metadata.

Our approach also requires the description of platform grammar rules and well known application schemas. Further work [Abd-el-Kader, 2006] should explore the mechanisms to acquire these descriptions.

## References

- [1] Gesbert, N.: Formalisation of Geographical Database Specifications. In: Proceedings of the ADBIS Conference on Advances in Databases and Information Systems (2004) 202-211.
- [2] Environmental Systems Research Institute, Inc.: Modeling our World, the ESRI guide to geodatabase design. ISBN 1-879102-62-5 (1999)
- [3] ISO TC21: ISO 19109 Geographic Information - Rules for Application Schema (International standard)(2005)
- [4] Bédard, Y., Visual modelling of spatial databases: Towards spatial PVL and UML. *Geomatica*, 53(2) (1999) 169-186
- [5] Parent, C., Spaccapietra, S., Zimányi, E.: Conceptual modeling for traditional and spatio-temporal applications. The MADS approach. Springer Verlag (2006)
- [6] Pinet, F., Kang, M.A, Vigier, F.: Spatial Constraint Modelling with a GIS Extension of UML and OCL: Application to Agricultural Information Systems. In: Proceedings of the Metainformatics International Symposium (2004)
- [7] Friis-Christensen, A., Christensen, J.V, Jensen, C.S : A Framework for Conceptual Modeling of Geographic Data Quality. In: P. Fisher (ed.): Proceeding of the 11<sup>th</sup> international Symposium on Spatial Data Handling (2004) 605-616
- [8] Leite, F.L., De Sousa, A.G., De Souza Baptista, C., Nunes, C.P, Da Silva, E.R, De Almeida, D.R, De Paiva, A.C: Migratool: Towards a Web-Based Spatial Database Migration Tool. In: Proceedings of the 16th DEXA conference (2005) 480-48.
- [9] Josselin, D.: Spatial data exploratory analysis and usability. *Data Science Journal (Spatial Data Usability Section)*, 2(26) (2003)
- [10] Abd-el-Kader, Y., Bucher, B.: Cataloguing GI Functions provided by NonWeb Services Software Resources Within IGN. In Proceedings of the 9th AGILE International Conference on Geographic Information Science, appendix (2006)
- [11] Lemmens, R., Granell, C., Wytzisk, A., de By, R., Gould, M., van Oosterom, P.: Semantic and syntactic service descriptions at work in geo-service chaining.. In: Proceedings of the 9<sup>th</sup> AGILE Conference on Geographic Information Science (2006) 51-61