Bédard, Y, S. Larrivée, M.J. Proulx & M. Nadeau, 2004, Modeling Geospatial Databases with Plug-Ins for Visual Languages: A Pragmatic Approach and the Impacts of 16 Years of Research and Experimentations on Perceptory, S. Wang et al. (Eds.): COMOGIS Workshops ER2004, LNCS 3289, pp. 17–30, 2004., Shanghai, China

Modeling Geospatial Databases with Plug-Ins for Visual Languages: A Pragmatic Approach and the Impacts of 16 Years of Research and Experimentations on Perceptory

Yvan Bédard, Suzie Larrivée, Marie-Josée Proulx, and Martin Nadeau

Centre for Research in Geomatics, Dept of Geomatics Sciences
Laval University, Quebec City, Canada
Tel.: 1-418-656-2131
Fax: 1-418-656-7411
{yvan.bedard,suzie.larrivee,marie-josee.proulx,
martin.nadeau}@scg.ulaval.ca

Abstract. Modeling geospatial databases for GIS applications has always posed several challenges for system analysts, developers and their clients. Numerous improvements to modeling formalisms have been proposed by the research community over the last 15 years, most remaining within academia. This paper presents generic extensions (called Plug-Ins for Visual Languages or PVL) to facilitate spatial and temporal modeling of databases. For the first time, we explain its intrinsic relationship with an extended repository and how it has been influenced by pragmatic lessons learned from real life projects. We describe how we use PVLs with UML and how 16 years of fundamental research, diverse experimentations and feedbacks from users over the world shaped our approach. The final section presents Perceptory, a free repository-based UML+ PVL CASE developed to improve geospatial database modeling.

1 Introduction

Modeling geospatial databases for GIS applications has always posed several challenges for system analysts, system developers as well as for their clients whose involvement into the development of such a project is not a familiar endeavor. To help solving this problem, numerous software and database modeling techniques have been proposed over the last 25 years. Over the last 8 years, UML has emerged as a standard and has been widely adopted by industry and academia, including the geomatics community. Amongst the characteristics of UML, there is a formal way to extend it with stereotypes identified by a textual or pictogram notation [16]. Using pictograms to model databases isn't new. In the field of cartography, it was first proposed in 1989 by Bedard and Paquette [7] to help E/R models depicting the geometry of cartographic features. This first solution has been tested in several projects (National Topographic Database of Canada, Montmorency Forest, Urban database, etc.) and enhanced over time to become Modul-R [5, 6]. Modul-R was an E/R-based solution supported by Orion, the first CASE including spatial pictograms and automatic code

S. Wang et al. (Eds.): ER Workshops 2004, LNCS 3289, pp. 17–30, 2004.

generation for commercial GIS [6,8]. In 1996, after several experimentations with real projects, an important shift was made to simplify the proposed solution while increasing its expressive power. Practical projects clearly indicated the need to better support unexpected complex geometric and temporal situations. On the other hand, Modul-R expressiveness was continuously underused by both practitioners and students in spite of previous training. Such common trend has psychological roots (ex. the concept of satisficing) and stems from projects constraints such as allowable time, allowable cost and client desire to see something running on his machine. In addition, many practitioners hesitated to abandon their familiar formalism to adopt Modul-R underlying MERISE E/R graphical notation. These practical experiences suggested realignment and led to improved pictograms, better balance of information between the schema and the dictionary, and formalism-independent PVLs (Plug-ins for Visual Languages). This was a major departure from other solutions which kept adding features to their unique graphical notation (sometimes proprietary) and didn't propose a formal repository to store textual information hardly suitable for graphical representation.

A first test was made in 1996-97 while designing the road network OO model for the government of British Columbia. In Summer 1997, the first version of the CASE tool Perceptory was running; it implemented the PVLs as UML stereotypes. In parallel, other researchers built slight variations of the pictograms, added extra ones or developed similar concepts [27, 28, 14, 25, 26, 17, 21, 29]. Fig. 1 from Filho and Lochpe [15] presents a historical overview of formalisms development.

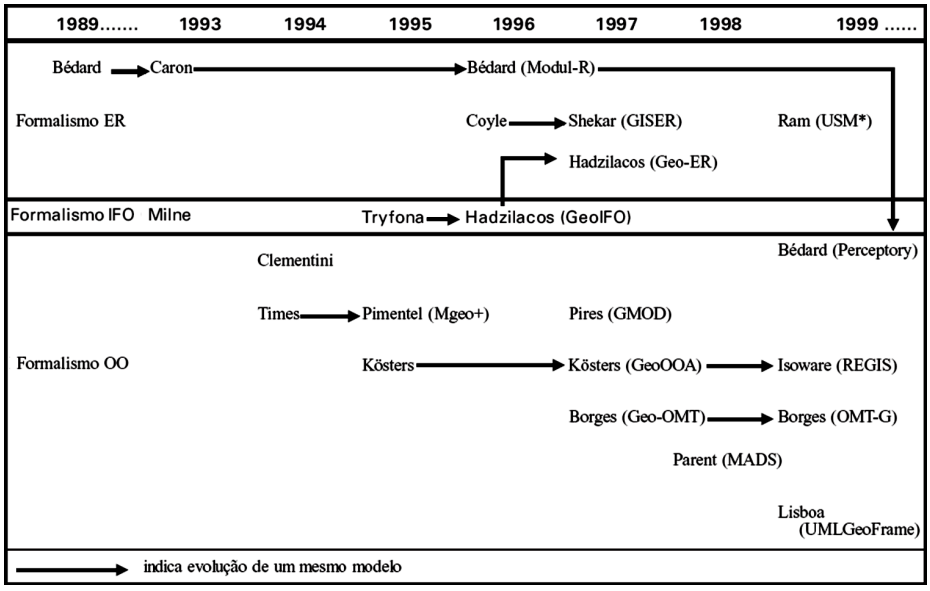


Fig. 1. Historical overview of spatial formalisms (published in 15).

It is beyond the scope of this paper to compare the different solutions, however the approach presented here differs from the others in two fundamental ways: a "plug-in approach" and a "fully-integrated schema + repository approach". The rational for the first difference is presented in [4]. The rational for the second differences is a reaction

to the leading tendency to perceive "modeling" solely as a schema exercise while in fact it is not; a schema without clear and complete explanations of its content is meaningless and its robustness cannot be validated. We have searched for the most efficient balance between information best represented in the schema and information best represented in textual form [6]. This has allowed us to avoid building too many pictograms (as we tended in the early years, prior practical experiences). This has also allowed us to define spatial and temporal extensions to traditional data dictionaries, a work that is unique amongst the solutions proposed. The proposed balance between graphical notation extensions and object dictionary extensions results from 16 years of research and numerous lessons from database designers in several organizations and projects such as the Canada Road Network, Canada Hydrographic Network, National Topographic Database, Quebec Multi-Scale Topographic and Administrative Data Warehouse, M3Cat Georeferenced Digital Libraries, Quebec Public Health, and many more involving parcels, facilities, natural resources, archeology, olympic athletes training, etc. Over 1000 undergraduate and graduate students have worked with the developed solution as well as GIS professionals in continuing education courses in Canada, France, Switzerland, Tunisia, Mexico and World Bank projects for Burkina-Faso and Senegal. Perceptory is downloaded over 300 times per month and its web site (http://sirs.scg.ulaval.ca/perceptory) is visited thousands of times yearly. Perceptory is used in over 40 countries. The feedback from this large base of users and projects contributes to the improvement of PVLs and repository extensions.

In the next section, we present the modeling approach we developed over the years. Pragmatic lessons combined with studies in cognitive science and psychology have led us to adopt a symbiotic approach that takes into account humans' abilities and limitations to create and read models and texts (analysts, developers, users), the constraints that apply to real-life projects (time, cost), the needed simplicity vs expressive power of the formalism, the need to be compatible with international standards, and the possibility to use other tools to support modeling than Perceptory.

2 Pragmatic Lessons

Experience has shown that database designers have difficulty to create a first model that reflects "what" users need, i.e. the intended content of the database, without interfering with implementation issues ("how" to provide it). Not separating properly conceptual and implementation issues pollutes the analysis process with irrelevant details and often hides the intentions of the clients. The emphasis of several UML books on syntax and grammar lead to underestimating the importance of going incrementally from technology-independent database analysis to technology-specific development. In order to help analysts stay at the conceptual level and to facilitate communication with his clients, we make schemas using only the essentials of objects' semantics, geometry and world time. Accordingly, we use only the required subset of the UML object-class model and details are stored in the repository in natural language or codes as needed (detailed semantics, geometric acquisition rules, existence and evolution rules, minimum dimensions, resolution, extent, integrity constraints, derivation rules, ISO data types, etc.). In fact, experience has led our team to adapt an agile approach to database modeling (see [1], [9], [12], [18] and [22]) and to make conceptual sche-

mas with a UML-Lite approach as opposed to [23]. These essentials include class, attribute without data type and visibility, operation without visibility, association with multiplicities (written consistently with two explicit numbers for min and max, no * nor single number) written next to its name to make reading easier, multiplicities for attributes, association class, generalization, aggregation, package, ad hoc stereotype, spatial properties of classes and attributes, temporal properties of classes and associations (existence) as well as temporal properties of attributes and geometries (evolution). Constraints on relationships are accepted as well as notes. On the other hand, since hundreds of spatial relationships between classes are possible in a schema, they are not included in the schema and are kept implicit (they are assumed to be derivable using GIS spatial analysis applied to the geometry of the classes) unless a user desires to put emphasis to some spatial associations for semantics reasons or integrity checking (ex. House IsOn 1,1 Lot). Spatial integrity constraints involve semantics and geometry (ex. Road cannot overlay Dam if Road Class is type 2 and Dam is shorter than 200m) and are typically kept outside of the schema since they are detailed information, they often vary according to attribute values and consequently may end up numbering over hundreds of times the number of classes. They are better managed in specialized tools such as CSory (Constraints-in-Space repositORY) which uses both the repository of Perceptory and the Erelate [13] and CRelate [11] ISO matrices to allow users to define their constraints based on geometry and semantics (objects and attributes). Doing so avoids polluting the schema and disturbing the focus of the model from clients' immediate objects of interest. Instead, an easily readable report is automatically produced. In fact, organizations don't create large models of spatial integrity constraints since it is practically impossible to cover a significant part of all cases, commercial software cannot check them automatically, and they rely on data providers for quality assurance. At best, they write the most important ones textually and CSory has been designed with this fact in mind. Such decision to explicitly exclude spatial integrity constraints that are not sensitive to the client from the database schema seems to be a unique approach among the above-mentioned solutions, but it likely is the only one based on real experiences with large topographic databases where strategies have to be defined to restrict constraints to a manageable number (ex. 400 out of 20000 possibilities for about 48 object classes [24].

Compatibility with international standards such as ISO/TC-211 and OGC is also commonly required [19, 20]. However, ISO jargon doesn't express all possible geometries (ex. alternate and facultative geometries) and they are not cognitively compatible with clients' conceptual view who assumes a topologically consistent world (ex. GMPoint vs TPNode, GMCurve vs TPEdge, GMSurface vs TPFace, Aggregate vs Multi). So, conceptual modeling is more efficient with PVLs. They can be mapped with standard ISO geometries automatically with Perceptory or with user assistance [10]. Doing so allows for automatic generation of code, reports, XML files and so on with ISO technical syntax and content (see lower left of Fig. 6). Similarly, there frequently is a linguistic need to support a single schema and dictionary (as well as reports) expressed in multiple languages (ex. French and English in Canada) in combination with a specific technical jargon (ex. French+OMG/UML, English+ISO; see upper-right of Fig. 6).

Another reality is the necessity for some analysts to continue working with their own formalism (sometimes, these are in-house variations of a known formalism) and

CASE tool they already have. This pragmatic requirement was behind the idea to develop a plug-in solution that can be integrated into any object-oriented and entity-relationship formalism, and to make the pictograms available in a special font (downloadable from Perceptory web site) which can be integrated into any software. PVLs can thus be installed and used in commercial CASE tools, word processing, DBMS, GIS, in-house software, etc. Extending CASE tools repository allow them to accept detailed information about the geometry of objects. We developed Perceptory to have a research tool we can modify at will for experimentation, to encourage teaching spatial database modeling by providing a free tool to academia worldwide and to help industry seeking for a more efficient technology specifically geared towards spatial databases. The next two sections present the PVL and Perceptory.

3 Using PVLs for Spatial Database Modeling

The PVL concept was introduced in 1999 by Bedard [4]. PVLs are simple but powerful sets of pictograms and grammar forming a graphical language used to depict any possible geometry. In database modeling, they are used for spatio-temporal properties of object classes, of attributes, of operations and of associations via association classes (see [4, 10] for more details). They offer a high level of abstraction by hiding the complexity inherent to the modeling of geometric and temporal primitives as well as implementation issues specific to each commercial GIS product or universal server. They encapsulate parts of the data structure dealing with geometric primitives, a technique called Sub-Model Substitution when introduced by Bedard and Paquette [7]. They are compatible with common practices to extend formalisms such as UML stereotypes (ex. relational stereotypes [23], multidimensional stereotypes [30]). We have developed two PVLs for 2D applications: spatial PVL and spatio-temporal PVL. Their equivalent 3D spatial PVLs are also available. The next sections present the use of PVLs in UML class diagrams for the modeling of 2D spatial databases, 2D multi-representations databases, 2D temporal and spatio-temporal databases, and finally 3D spatial databases.

3.1 Using PVL for 2D Spatial Database Modeling

This PVL was the first developed and it by far the most largely used as it suffices for most applications. It is composed of only three spatial pictograms and three special pictograms to keep the schema simple and to facilitate communication with customers. The geometric details are written in the repository. These pictograms are made of shapes (0D, 1D, 2D) within a 2D square which represents the 2D dimension of the system universe (). Simple point geometry pictogram can be used for object classes like "Fire hydrant", simple linear geometry for object classes like "Road segment centerline" and simple surface geometry for object classes like "Park". Using no pictogram indicates the objects of a class are not going to be represented on a map. All forms of pictograms that represent the object geometry are presented in Table 1.

Table 1. Possibles PVL forms for 2D spatial database modeling of object geometry.

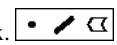
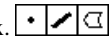
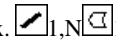
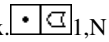
Grammar	Notation	Description and examples
Simple geometry		
One pictogram with implicit 1,1 default multiplicity (not written).	Ex.  Ex.  Ex. 	Every instance is represented by one and only one simple geometry. Ex. A 'Fence' is represented by one and exactly one 1D geometry.
Aggregate geometry (AND)		
Simple aggregate has a simple pictogram with 1,N multiplicity.	Ex.  1,N Ex.  1,N	Each instance is represented by an aggregate of shape of *same dimension.* Ex. An 'Orchard' is an aggregate of points (not a polygon).
Complex aggregate has different shapes in the same pictogram where shape order has no incidence.	Ex.  Ex.  Ex. 	Each instance is represented by an aggregate of shape of *different dimensions.* Ex. 'Large rivers' are composed of lines and polygons. *An aggregate of aggregates is an aggregate (i.e. no need of 1,N multiplicity).*
Alternate geometry(XOR)		
Two or more pictograms on the same line.	Ex.  Ex.  Ex.  1,N1,N Ex.  1,N	Each instance is represented by a geometry or another but not both (the exclusive OR). The proposed geometries can be simple and/or aggregates. Ex. 'Building' can be represented by a point if it is smaller than a fixed value or by a surface if it is larger.
Multiple geometries		
Two or more pictograms on different lines	Ex.   Ex.  1,N 	Each instance is represented by each geometry but usually only one is used at a time. Ex. A 'Town' may have two geometries, a surface delimited by its boundary and a point representing downtown. *All geometry variations can be used within multiple geometries.*
Complicated geometry		
Exclamation mark pictogram *Rarely used as it is facultative*		When the analyst feels that although it is possible, expressing the geometry with previous pictograms becomes too complicated. Description is in the repository. Ex. Bus Network made of complex aggregations of roads (multiple lines) + bus stops (points or polygons) + parking lots (single or multiple polygons).
Any possible geometry		
Wildcard pictogram *Rarely needed*		All geometries are possible, without restriction. Ex. Historical Feature as a point (ex. statue), line (ex. historic street), polygon (ex. park), polygon + line (ex. historic place with adjacent streets), set of polygons (ex. group of buildings), etc.
Temporarily undefined geometry		
Question mark pictogram. *Rarely used.*		The geometry is temporarily unknown. Will be replaced by meaningful pictograms.

Table 1. (Continued).

Grammar	Notation	Description and examples
Facultative geometry		
Add a minimum multiplicity of 0 and a maximum multiplicity after the pictogram.	Ex. 0,N Ex. 0,1 Ex. 0,1	Only certain instances have a geometry. For example, all buildings are in the database, but only public ones have a position and shape and will appear on maps. *All the above geometry variations can be facultative.*
Derived geometry		
Italicize the pictogram (to remind the slash used in UML).	Ex. Ex. Ex. Ex.	The geometry is obtained from the processing of other geometries or attributes. Ex. Country polygons can be derived from state polygons. *All the above geometry variations can be derived, each pictogram independently from the other.*

For object classes, we place the pictograms on the left side of their name. When applied to attributes, pictograms indicate that the value of an attribute varies within the geometry of an object (an attribute with different values in different places for a single conceptual object). This avoids having to create at the conceptual level a new object class which would be meaningless for the client. For example, using next to the attribute "number of lanes" of the class "Road" (see Fig. 2) is cognitively compatible with the client perception of "number of lanes" as a characteristic of Road that varies spatially within an occurrence. Implementation can either create sub-segments or use GIS dynamic segmentation.

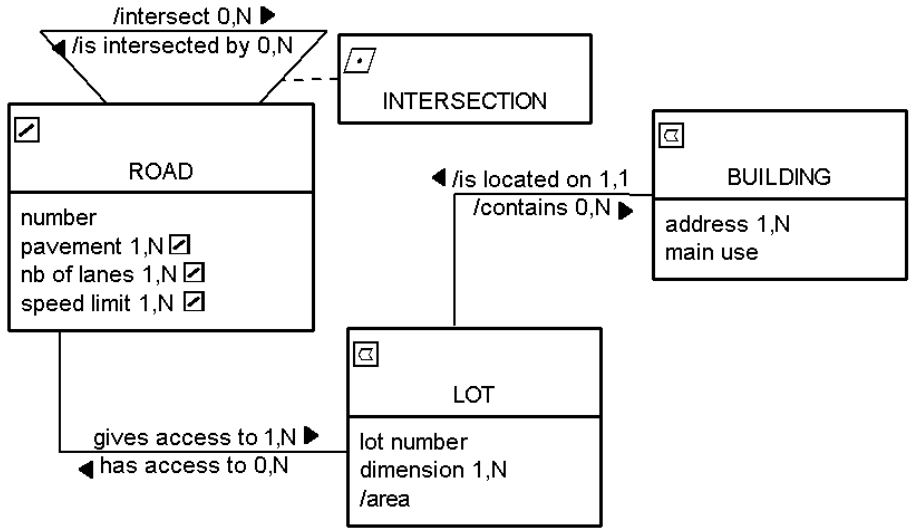


Fig. 2. Spatial schema showing 2D pictograms for object classes and attributes, two spatial associations derived from object geometries, one derived geometry for an association class, one spatially non-derivable association.

Concerning the modeling of spatial relationships, we include only spatial relationships that are semantically significant or sensitive for the customer to avoid overloading the diagram. In fact, spatial relationships of immediate interest to the user will often be the ones implemented explicitly, typically being derived from GIS processing and translated into foreign keys or tables during code generation. Such meaningful spatial relationships are modeled as derived associations (see UML "/" syntax in Fig. 2). Derivation details are explained in the repository.

Pictograms can also represent the result of a spatial process as for Intersection class in Fig. 2 where the derived geometry of the association class is the result of the spatial intersection association. Details of the processing are in the repository.

3.2 Using PVL for Multi-representations Database Modeling

The PVL already supports multiple geometries and is immediately usable for multi-representations databases. In addition, it also allows expressing generalization processes to build such multiple representations. Generalization processes are expressed as operations in OO models, i.e. operations on the geometry of an object class. Of the four possible generalization results presented hereafter, the last three lead to storing multiple representations:

- automatic on-the-fly generalization;
- automatic generalization storing the resulting geometry for performance reasons;
- semi-automatic generalization storing results to avoid repeating the operation;
- manual generalization and storing of the results.

One could add an additional way to create multiple representations: new acquisition.

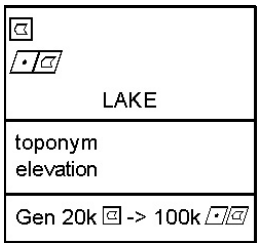


Fig. 3. Example of generalization operation and multiple representations of an object class.

In the previous example taken from the Quebec Topographic Database, a lake has an original polygonal geometry at a map scale of 20K and a derived alternate point/polygonal geometry at the 100K scale. When resulting geometries (ex. 100K) are stored in the database, we then add them *in italic* as multiple representations. Examples and details can be found in [3].

3.3 Using PVL for Temporal and Spatio-temporal Database Modeling

The spatio-temporal PVL is used to manage real world time (valid time, measured or simulated) and to indicate the user want to access it on-line (sometimes called "long

transactions"), i.e. to build a temporal database. The spatio-temporal PVL is defined by 0D or time instant ( ) and 1D or time interval ( ) pictograms which are used for class existence, descriptive evolution and geometric evolution.

For a class existence, pictograms are located on the right side of its name. They illustrate if the life of every occurrence is considered instantaneous or durable. For example, can be used for classes having instantaneous existence like "Accident", and can used for classes having durable life like "Building". For evolution, we use if attribute values or geometry are valid only for an instant and if they are valid for longer periods. "Temperature" is an example of instant validity when new values are obtained every 15 minutes and the temporal granularity of this field or of the database also is 15 minutes. “Commercial value” is an example of attribute with durable values. Selecting between or depends on the temporal granularity defined into the repository for each class, attribute and geometry.

All the forms presented for 2D spatial pictograms can be applied to temporal pictograms. For example, complex aggregated temporality is represented by , simple aggregate by 1,N, "any possible temporality” by , complicated temporality by and unknown temporality by . Alternate temporality is represented by many pictograms with or without multiplicity on the same line while they are on different lines for multiple temporalities. No user interest into keeping trace of object existence leads to use no temporal pictogram next to the name of the class. Similarly, no user interest to keep trace of past/future attribute values and geometries lead to no temporal pictogram next to them.

The default multiplicity for class existence is 1,1 because objects have only one life (if well defined). The evolution of an object being the aggregate of its states, its default multiplicity is 1,N since pictograms indicate a client's interest to keep several states of the attribute or geometry it is applied to. Typically, most occurrences will have time to evolve during their life (the N multiplicity), but some won't with regard to a given attribute or geometry due to their short life or stability (the 1 multiplicity). Fig.4 shows a Building having a durable existence (one and only one life), durable states for its attribute “commercial value” and no need to keep track of past addresses.

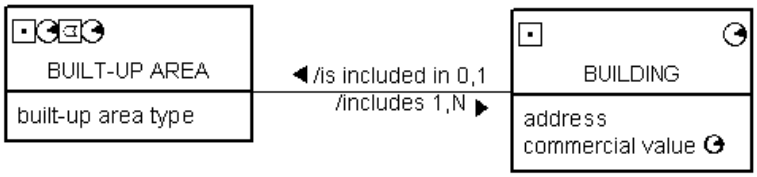


Fig. 4. Example of existence, descriptive and geometric evolutions.

Geometric evolution of objects involves shape modifications and/or displacements. A temporal pictogram is then placed at the right of the spatial pictogram (after the multiplicity if it has one). In the case of an alternate geometry, each spatial pictogram has its own temporal pictogram. Fig.4 illustrates the geometric evolution of the Built-Up Area of the Canadian National Topographic Database which occurrences are a point if smaller than 250 000m² or an area if greater. This area may evolve and the point move or become an area if the surface becomes greater than 250 000 m². These rules are indicated in specific fields of the repository.

3.4 PVL to Design 3D Spatial Database

Modeling 3D spatial databases is more complex than modeling 2D databases. Several objects of a 2D universe can be translated for a 3D universe simply by deriving their elevation from a DTM (Digital Terrain Model) or by extruding their height from an attribute. One may also have true 3D objects. To represent these nuances adequately, pictograms are made of shapes within a 3D cube instead of a 2D square in order to represent the three dimensions of the system universe. Second, to obtain the shape of the objects, we transpose each shape of the 2D spatial pictograms in a 3D space in a way preserving the ground trace (2D view) and giving an elevation or not to them. We obtain 6 pictograms as illustrated in Fig 5, where:

- a 0D object ( )in a 2D universe becomes a 0D object ( ) in a 3D universe if it is an object without thickness or a vertical 1D object ( ) if it has a height;
- a 1D object ( ) in a 2D universe becomes a 1D object ( ) in a 3D universe if it is an object without thickness or a vertical 2D object ( ) if it has a constant or variable thickness;
- a 2D object ( ) in a 2D universe becomes a 2D object ( )in a 3D universe if it is an object without thickness that embraces or not the digital terrain model or becomes a 3D object or volume ( ) if it has a constant or variable thickness;

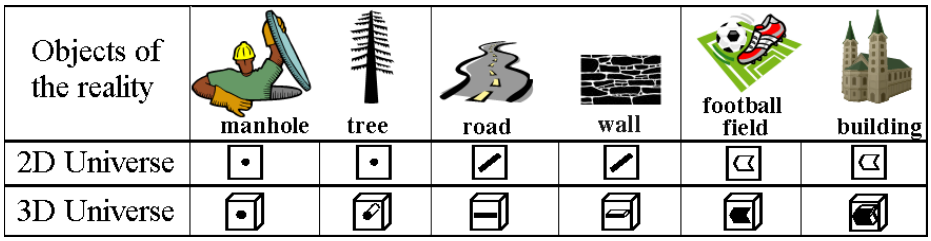


Fig. 5. Similarity between 2D and 3D pictograms to represent real world objects.

All variations described for 2D pictograms in Table 1 can be applied to 3D pictograms. For example, a complex aggregate , an alternate geometry , an aggregation of volumes 1,N and a complicated geometry .

4 Perceptory

Perceptory is a free CASE tool developed as a MS-Visio stencil for spatial database modeling (although it may also serve for non-spatial databases). It is based on UML class diagram and the PVL identified earlier. The name *Perceptory* comes from *perception*, referring to the process of phenomenon representation, and *repository*, for knowledge documentation. It is supported by a rich web site [2]. This CASE tool has a rich multi-language (actually French, English and Spanish) and multi-standard repository (OMG-UML, ISO/TC 211 19115, ISO/TC 211 19110) where schema and forms can switch on-the-fly between languages as needed once the user has translated

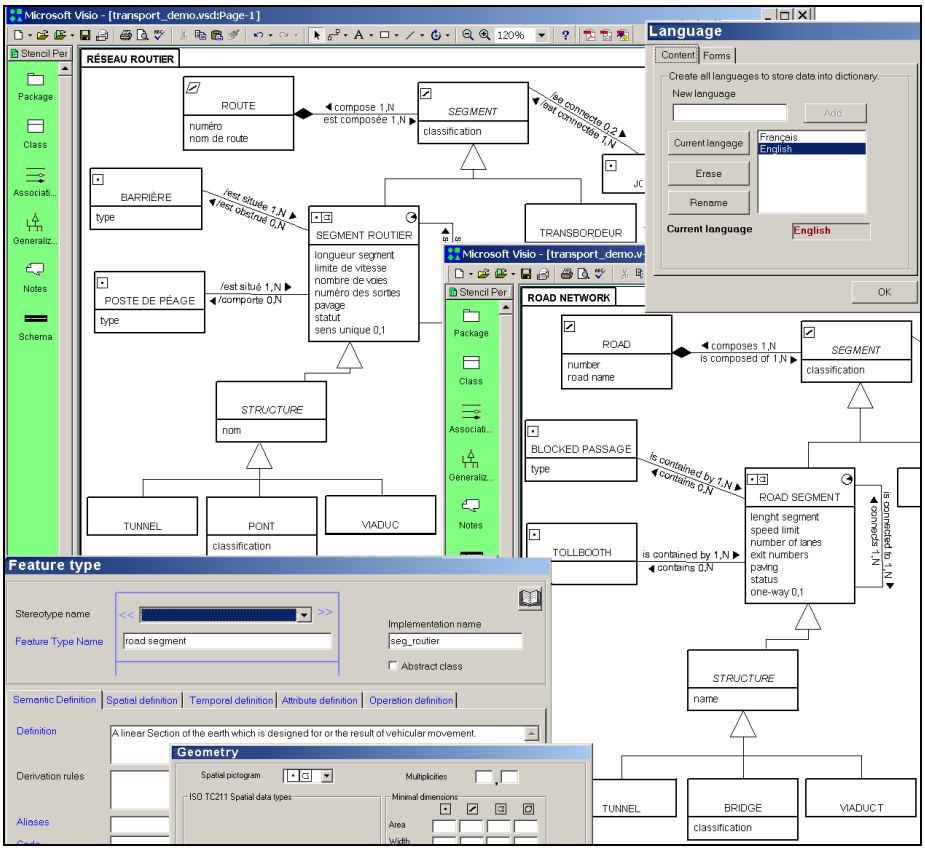


Fig. 6. Perceptory multistandard and multilanguage interfaces and schema viewing.

text fields. A same form can also be viewed in many languages simultaneously. Fig. 6 shows schemas in French and English synchronized thru the same repository. The larger form in the lower left shows ISO-TC211 19110 terms in blue for object definition while the smaller one shows the geometric characteristics of a class.

Perceptory has a utility to generate automatically the database code skeleton for commercial GIS and universal servers. SQL commands or database structures proper to each product are generated as well as log files and mappings between conceptual and implementation views. This process is user-assisted and allows the developer to select optimization strategies.

There is a report generator to print user-selected parts of the repository according to a user-defined content template or a predefined one (ex. standard ISO TC211 19110 content). Pictures or drawings that have been inserted into Perceptory repository during the design of the database are included (ex. to print topographic data acquisition specifications). The report can be produced in all the languages used in the repository in MS-Word and XML formats. The following table shows how Perceptory adheres to the philosophy we described earlier with regard to balancing information load between the schema and the repository.

Table 2. Balancing the information in the schema and in the repository for the main elements.

Modeling elements ...	**... applied to...**									**Info in**	
	Package	Class	Association	Attribute	Operation	Spatial picto	Temporal picto	Domain	Metadata	Model	Repository
UML notation (shape/text)	X [3]	X [3]	X [1,3]	X [3]	X [3]			X [2]		X [3]	
Model information									X	X [4]	X
Element name	X	X	X	X	X			X		X	X
Semantic definition	X	X	X	X	X			X			X
Implementation name		X [6]	X [6]	X [6]	X [6]			X [6]			X [6]
Derivation rule		X	X	X		X	X			X [5]	X
Stereotype name		X								X	X
Abstract class		X								X	X
PVLs		X	X	X	X					X	X
Multiplicity			X	X		X	X			X	X
Minimal dimension						X	X				X
Acquisition rule						X	X				X
Reference system							X		X		X
Coverage							X		X		X
Data format				X		X	X				X
Formal language					X						X
Role name and constraint			X							X	X
Association constraint			X							X	X
Etc.											

[1] *Topological associations are usually not modeled*
[2] *Enumerated domains only*
[3] *Must be created in the schema*
[4] *Key elements only*
[5] *Via UML slash (/) or italic pictogram*
[6] *If entered by choice during analysis*

5 Conclusion

In a field such as database design which involves people, communication skills and technical knowledge, theoretical research cannot succeed without thorough experiences. Over the years, such experiences has alerted us that going after the theoretically most complete and rigorous language has to be balanced with human capabilities, tools functionalities and project constraints. Taking these considerations into account leads to a more complete research, one buildings on their symbiosis to deliver results in sync with their fundamental raison d'être: being used to improve spatial database design. The proposed solution has evolved over time and resulted in unique concepts: (1) PVLs made of pictograms that constitute a powerful yet simple language of their own and which may be used for several purposes and in different tools; (2) a global design balancing and testing both schema representations and textual representations in a repository; (3) a clear separation of the tasks involved in conceptual modeling from the ones involved in defining spatial integrity constraints, while keeping the two in sync. In particular, balancing the expressive power of PVL with

that of textual explanations in a repository to describe the thousands of potential geometric and temporal characteristics combinations of an object is a challenge forever. So is balancing the readability and usability of PVLs with conceptual fidelity, robustness, completeness and, one must admit, some personal taste. All this research has also been influenced by pragmatic lessons which, on the overall, have led us to adopt a more agile approach to database design (in system development sense, see [1], [9], [12], [18], [22]).

Future research will involve a more global approach, a repository-based approach where Perceptory will become a UML front-end to enter and visualize database design components. Such expanded repository, called ISTory (Integrated Spatial and Temporal repositORY), will be the heart of a family of tools sharing common concepts: ontology engine, code generators, report generators, spatial integrity constraints, data cubes, catalogs, etc. ISTory metastructure will contain information to facilitate semantic interoperability, data quality analysis, transactional and analytical database development, language and standard translating, etc. The result of such endeavor, which builds on past and ongoing projects, will be made accessible as an XML file and provide a unique, powerful solution in sync with today's trends such as the semantic web and interoperability.

Aknowledgements

The authors wish to acknowledge the financial support of Canada Natural Sciences and Engineering Research Council, Laval University, and major testing partners such as the Ministry of Natural Resources Canada and the Ministère des Ressources Naturelles, de la Faune et des Parcs du Québec. We are thankful to the other project partners, to the users who send us feedbacks and requests, and to the three anonymous reviewers.

References

1. Ambler, S.: Agile Model-Driven Development with UML 2.0. Wiley & Sons, NY (2004)
2. Bédard, Y., Proulx, MJ.: Perceptory Web Site, http://sirs.scg.ulaval.ca/Perceptory/ (2004)
3. Bédard Y, Proulx MJ, Larrivée S, Bernier E: Modeling Multiple Representation into Spatial Datawarehouses: A UML-based Approach, ISPRS WG IV/3, Ottawa, July 8-12 (2002).
4. Bédard, Y.: Visual Modelling of Spatial Database towards Spatial PVL and UML, Geomatica, 53(2), (1999) 169-185
5. Bédard, Y., Caron, C., Maamar, Z., Moulin, B., Vallière, D.: Adapting Data Model for the Design of Spatio-Temporal Database. Comp. Env. and Urban Systems, 20(l) (1996) 19-41.
6. Bédard, Y., Pageau, J., Caron, C.: Spatial Data Modeling: The Modul-R Formalism and CASE Technology. ISPRS Symposium, Washington, August 1-14 (1992)
7. Bédard, Y., Paquette F.: Extending entity/relationship formalism for spatial information systems, AUTO-CARTO 9, April 2-7, Baltimore (1989) 818-827
8. Bédard, Y., Larrivée, S.: Développement des systèmes d'information à référence spatiale: vers l'utilisation d'ateliers de génie logiciel. CISM Journal ACSGC, 46(4) (1992) 423-433
9. Boehm, B., Turner, R.: Balancing Agility & Discipline. Addison-Wesley, NY (2004) 304 p.

10. Brodeur, J., Bédard, Y., Proulx, MJ.: Modelling Geospatial Application Database using UML-based Repositories Aligned with International Standards in Geomatics, ACMGIS, November 10-11, Washington DC, (2000) 36-46
11. Clementini, E., Di Felice, P., Van Oosterom, P.: A small set of formal topological relationship suitable for end users interaction. Third symposium on Large Spatial Database, No. 692, Singapore, Springer-Verlag, NY (1993) 277-295
12. Cockburn, A.:. Agile Software Development. Addison-Wesley, NY (2002) 278 p.
13. Egenhofer, M., Herring J.: Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases, Tech. Report, University of Maine (1990) 28 p.
14. Filho, J.L., Iochpe, C.: Specifying analysis patterns for geographic databases on the basis of a conceptual framework. ACMGIS, Vol. 7, Kansas City, USA. (1999) 7-13
15. Filho, J. L., Iochpe, C.: Um Estudo sobre Modelos Conceituais de Dados para Projeto de Bancos de Dados Geográficos, Revista IP-Informática Pública, 1(2) (1999) 37-90
16. Fowler, M.: UML 2.0, CampusPress (2004) 165 p.
17. Hadzilacos T., Tryfona N.: An Extended Entity-Relationship Model for Geographic Applications. SIGMOD Record, 26 (3) (1997)
18. Highsmith J.: Agile Software Development Ecosystems. Addison-Wesley, (2002) 448 p.
19. ISO/TC211 19110, Geographic information: Methodology for feature cataloguing (2004)
20. ISO/TC211, 19115, Geographic information: Metadata (2003) 140 p.
21. Kosters, G, Pagel, B., Six, H.: GIS-Application Development with GeoOOA. IJGIS, 11(4) (1997) 307-335
22. Larman, C.: Agile & Iterative Development. Addison-Wesley (2004) 342 p.
23. Naiburg EJ., Maksimchuk, RA.: UML for Database Design, Addison-Wesley (2001) 300 p.
24. Normand, P., Modélisation des contraintes d'intégrité spatiale : théorie et exemples d'application, Ms. Degree, Dept. Geomatics Sciences, University Laval, 1999
25. Parent, C, Spaccapietra, S, Zimanyi, E., Donini, P.: Modeling Spatial Data in the MADS Conceptual Model. Int. Symp. on Spatial Data Handling, Vancouver (1998) 138-150
26. Parent C, Spaccapietra, S., Zimanyi, E.: Spatio-Temporal Conceptual Models: Data Structures + Space + Time, 7th ACMGIS, GIS'99, Kansas City, (1999) 26-33
27. Shekhar, S., Vatsavai1, R.R., Chawla, S., Burk, T. E.: Spatial Pictogram Enhanced Conceptual Data Models and Their Translation to Logical Data Models, ISD'99, Lecture Notes in Computer Science, Vol 1737, Springer Verlag (1999) 77-104
28. Shekhar, S., Chawla, S.: Spatial Databases A Tour, Prentice Hall (2003) 262 p.
29. Tryfona, N. Price, R., Jensen, C.S. Conceptual Models for Spatio-temporal Applications. Spatio-Temporal Databases: The CHOROCHRONOS Approach 2003, (2003) 79-116
30. Priebe, T., Pernul, G.: Metadaten-gestützer Data-Warehouse-Entwurf mit ADAPTed UML, 5th Int.Tagung Wirtschaftsinformatik (WI 2001), 19.-21. September, Germany (2001)