A CASE Tool for Geographic Database Design Supporting Analysis Patterns

Jugurta Lisboa F., Victor de Freitas Sodré, Jaudete Daltio, Maurício Fidelis Rodrigues Júnior, Valério Vilela

Department of Informatics, Federal University of Viçosa
36570-000, Viçosa, MG, Brazil
{jugurta, vsodre, jdaltio, mfrj, vvilela}@dpi.ufv.br

Abstract: This paper describes the development of an open source CASE tool, the ArgoCASEGEO, and its modular architecture. The ArgoCASEGEO allows the geographic database modelling based on the UML-GeoFrame conceptual model that is specific for Geographic Information Systems applications. The data dictionary associated to the modelled schema is stored as a XML/XMI document, aiming its use by other software. The ArgoCASEGEO follows a design methodology based on a reusable collection of analysis patterns. The analysis patterns collection is stored in a data base composing a Catalog. The catalog is attached to the ArgoCASEGEO. Thus, searching for existing analysis patterns will be an easier and efficient task.

Key words: Geographic database design, Analysis Patterns, CASE tool.

1 Introduction

Geographic Databases (GeoDB) are collections of geographically referenced data, manipulated by Geographic Information System (GIS). In the Geoprocessing area, normally the user itself is the one who develops the GIS applications. Thus, redundancy and inconsistency are strong characteristics in the majority of the GeoDB, many times compromising the system reliability and, consequently, putting great public or private investments into risk. For that matter, the development of methodologies and tools that assist the GeoDB designers are essential to improve the quality of GIS applications.

A GeoDB should be designed following a database project methodology that includes conceptual, logical and physical design phases [5]. To elaborate the data schema in the conceptual phase, a data model must be chosen. Various models for GeoDB have been proposed in the past years as GeoOOA [12], MADS [19], OMT-G [2], UML+SpatialPVL [1] and UML-GeoFrame [14].

At this point, reuse mechanisms may help less experienced designers through instruments that allow software components reuse by patterns definitions. Analysis

Patterns is a pattern category, which has been treated as a reuse instrument for requirement analysis and conceptual modelling [6], [7], [9], [11] and [20]. Analysis patterns permit reuse in a higher level than object-oriented class specialization because it makes possible to reuse a part of a data schema instead of a single class.

Concluded the conceptual modelling, the next step - logical design - consists of the conceptual schema transformation into a data schema compatible with the data model of the GIS that will be used. This stage of a conceptual schema transformation into a logical-spatial schema, and its settlement in a GIS, can be made automatically by a CASE (Computer Aided Software Engineering) tool. Some of these conceptual models previously mentioned are supported by CASE tools, for example, Perceptory [1], REGIS [10], AIGLE [13] and Publisher Java MADS [19].

This paper describes the ArgoCASEGEO architecture, an open source CASE tool for GeoDB modelling that supports the UML-GeoFrame model [14]. The conceptual schema elaborated by this tool is stored in XML (eXtensible Markup Language) format, so can be easily accessed and used. This tool also provides an automatic generation module, able to generate data schemas to the most common formats usually found in commercial GIS. Moreover, the ArgoCASEGEO has a support for reuse based on analysis patterns [7] through the Analysis Patterns Module that implements a Manager and a Catalog. Further information about the using advantages of analysis patterns in GeoDB conceptual modeling can be seen in [15] and [16].

Section 2 presents the UML-GeoFrame Model whereas section 3 details the development of the ArgoCASEGEO tool, showing its architecture and describing each module. Finally, section 4 brings final considerations and future works.

2 The UML-GeoFrame Model

The conceptual modelling of GeoDB based on the UML-GeoFrame model [14] produces an easy understanding conceptual schema, improving the communication between designers and/or users. Besides being used in the database schema elaboration, the UML-GeoFrame model is appropriate to the analysis patterns specification.

The GeoFrame is a conceptual framework that supplies a basic class diagram to assist the designer on the first steps of the conceptual data modelling of a new GIS application. The mutual use of the UML class diagram and the GeoFrame allows the solution of the majority requirements of GIS applications modelling. A geographic conceptual schema built based on the UML-GeoFrame model includes, for example, the spatial aspects modelling of the geographic information and the difference between conventional objects and geographic objects/fields. The specification of these elements is made based on the stereotypes set shown in Figure 1.

The first stereotype set (Geographic Phenomenon and Conventional Object) is used to differ the two main object types belonging to a GeoDB. The Geographic Phenomenon class is specialized in Geographic Object (◬) and Geographic Field (◬) classes, according to two perception ways of the geographic phenomena, described by Goodchild [8]. Non-geographic Objects are modeled on traditional form and are identified through the stereotype (△).

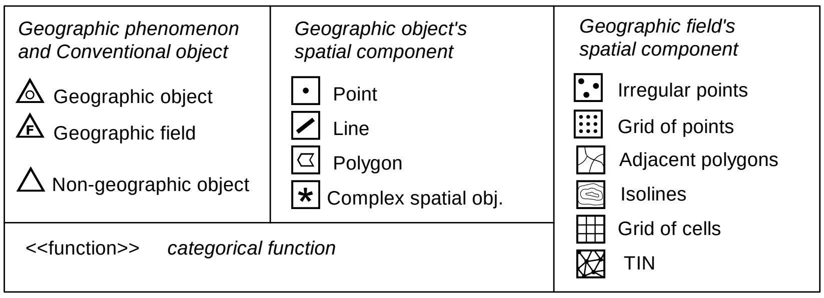


Fig. 1. Stereotypes of the UML-GeoFrame Model

The Geographic Object's Spatial Component and Geographic Field's Spatial Component stereotypes sets are used to model the phenomena spatial component according to object and field visions, respectively. The existence of multiple representations is modeled through the combination of two or more stereotypes on the same class. For example, a County class can have two abstraction ways of its spatial component, punctual and polygonal, that is specified by the stereotype pair (⊡ ⊡).

Finally, the stereotype <<function>> is used to characterize a special type of association that occurs when modelling categorical fields. According to Chrisman [3], in a structure of categorical covering the spatial is classified in mutually exclusive categories, that is, a variable has a value of category type in all the points inside a region. Figure 2 exemplifies the UML-GeoFrame model use showing a class diagram containing two themes: Education and Environment.

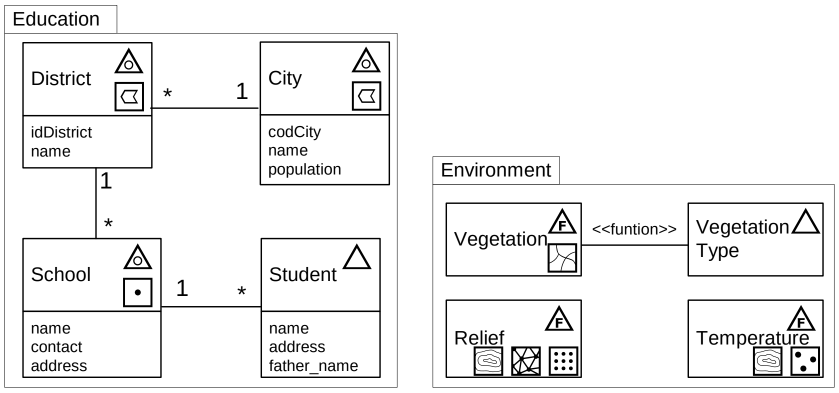


Fig. 2. An UML-GeoFrame schema example

The Education theme, modeled as a UML package, includes three geographic phenomena classes perceived in the object vision (District, City and School), and the Student class that is a non-geographic object. In the Environment theme, three classes of geographic phenomena perceived in the field vision are modeled, Vegetation, Relief and Temperature, each one with its different types of spatial representation. This theme still includes the Vegetation Type class, which is modeled as non-geographic object, being associated to the Vegetation class through the stereotype <<function>>, that is, each polygon is associated to a vegetation type.

3 The ArgoCASEGEO tool

ArgoCASEGEO is a CASE tool whose goal is to give support to the GeoDB modelling based on the UML-GeoFrame model. The data schemas elaborated using this tool are stored in XMI (XML Metadata Interchange) format, a syntax for conceptual schema storage, in XML documents [18].

XMI combines the definition, validation and sharing document formats benefits of XML with the specification, distributed objects and business-oriented models documentation and construction benefits of the UML visual modelling language.

A CASE tool is primarily a graphical drawing software. To avoid a great programming effort in developing a new graphical drawing tool, some existing graphical softwares were selected to be used as starting point. This software must support the UML class diagram drawing and be extensible to support the stereotypes defined in the UML-GeoFrame model.

After analyzing some options, the ArgoUML publisher was chosen as the base tool. Thus, the ArgoCASEGEO was developed as an ArgoUML software extension, a modelling tool found over a use license and open source distribution, developed in Java. Figure 3 illustrates the five-module-architecture of the ArgoCASEGEO.

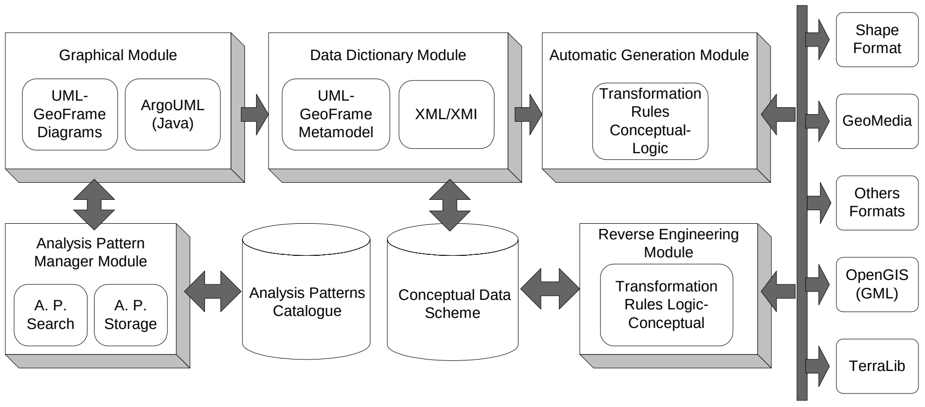


Fig. 3. The ArgoCASEGEO Tool Architecture

The Graphical Module allows the design of the conceptual schema, providing a set of constructors of the UML-GeoFrame model. The Data Dictionary Module stores the description of the diagram elements created by the designer. The Automatic Generation Module allows the transformation of the conceptual schema stored in the data dictionary into a logical schema corresponding to some models used in commercial GIS. The Analysis Patterns Catalog and its manager are defined in the Analysis Patterns Module. And finally, the Reverse Engineering Module, not yet implemented, will enable the designer to get conceptual schemas from existing GIS applications. The following sections describe these modules giving further details.

3.1. Graphical Module

The ArgoCASEGEO tool enables creation of diagrams that contain the constructors and stereotypes suggested by the UML-GeoFrame model. From this diagram the user can create its conceptual schema. An UML-GeoFrame conceptual schema supports three distinct class types: Geographic Object, Non-geographic Object and Geographic Field. The existing fields in the implemented class have name, attributes, operations and symbols corresponding to the spatial representation type (stereotypes).

These classes can be related by relationships as generalization & specialization, aggregation, composition or association. In an association, the relationship name and the multiplicity of each class can be specified. The classes can be grouped to form a definitive theme, which is modeled by the UML's Package constructor. Figure 4 illustrates the ArgoCASEGEO environment.

An UML-GeoFrame data schema can be saved as a new Analysis Pattern, which can be (re) used in new data schema, composing thus, the Analysis Patterns Catalog. On the other hand, if the designer is starting a new project it would be interesting to look up in the catalog in order to find existing analysis patterns.

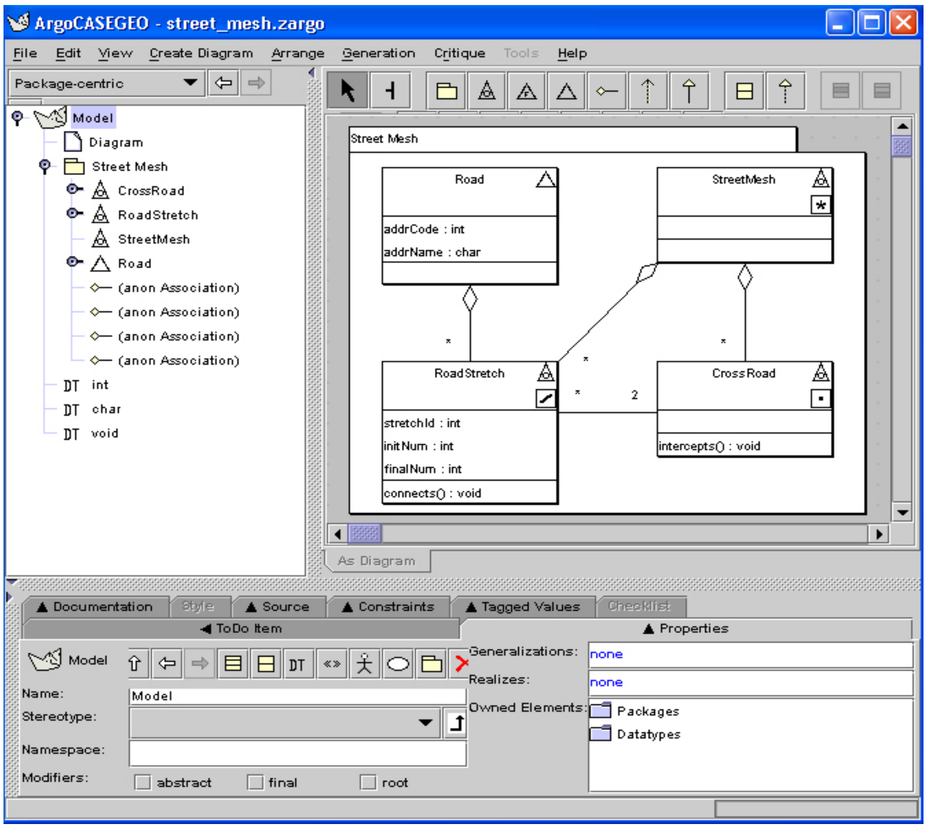


Fig. 4. The ArgoCASEGEO's graphical environment representing the analysis pattern Urban Street Mesh in the UML-GeoFrame model

3.2. Data Dictionary Module

The dictionary stores the data schema created by the user. A schema has two data types, the graphical data (drawing) and the semantic data (classes' names, attributes, associations' multiplicities, etc). The semantic data are stored in the data dictionary, while the graphical data are stored in an ArgoUML file. The data dictionary stores the conceptual schema in XMI format. Every class is delimited by a tag that contains the class name, its spatial representations and its features. The feature tag has two sub levels corresponding to the attributes and operations storage.

Figure 5 exemplifies the data dictionary to the Road Strech class (modeled in figure 4) whose spatial representation is line type, the direction and idStrech attributes. The types used in this definition, including the attribute type, parameters and operations' returned values are defined by the ArgoUML.

```
<Foundation.Core.GeographicObject>
   <Foundation.Core.ModelElement.name>RoadStrech
   </Foundation.Core.ModelElement.name>
      <Foundation.Core.GeneralizableElement.isLine
      xmi.value="true"/>
      <Foundation.Core.Classifier.feature>
         <Foundation.Core.Attribute>
            <Foundation.Core.ModelElement.name>direction
            </Foundation.Core.ModelElement.name>
            <Foundation.Core.Classifier xmi.idref="xmi.16"/>
         </Foundation.Core.Attribute>
         <Foundation.Core.Attribute>
            <Foundation.Core.ModelElement.name>idStrech
            </Foundation.Core.ModelElement.name>
            <Foundation.Core.Classifier xmi.idref="xmi.14"/>
         </Foundation.Core.Attribute>
      </Foundation.Core.Classifier.feature>
</Foundation.Core.GeographicObject>
```

Fig. 5. An UML-GeoFrame class in XMI representation

A specific tag that contains its name, related properties (that vary according to the type, association, aggregation or composition), its multiplicity and the classes' references that participate in the relationship, marks off the relationships between the classes modeled in the schema. From the generalizations definition vision, the internal tag is responsible for storing references to subclasses and super classes. Multiple inheritances are allowed. Finally, the package definitions are kept in a more external tag that includes everything previously described and has only its name as attribute.

3.3. Automatic Generation Module

After the conceptual modelling, the user needs to transform the elaborated schema into an effective implementation, characterizing a GIS application. As each GIS has its own data logical model, it is not possible to establish a single set of transformation rules to make the automatic generation of the logical-spatial schema. Thus, for each GIS the ArgoCASEGEO tool needs a specific Automatic Generation Module (AGM).

Two AGM have already been implemented in the ArgoCASEGEO tool. The first module transforms UML-GeoFrame schema to Shape format, used in the GIS ArcView [17]. A second AGM, described in the section below, transforms conceptual UML-GeoFrame schema into logical-spatial schema of the GIS GeoMedia.

3.3.1. GeoMedia Automatic Generation Module

The AGM-GeoMedia has as input the data dictionary identification that contains the conceptual schema to be transformed. To create a work environment in this software a connection with an existing database must be established. This connection will store the layers and the associated tables. For that matter, the AGM-GeoMedia creates a database Access (.mdb) to store all the elements generated by the automatic mapping.

For each element of the conceptual schema a specific transformation rule is applied. These rules are described as following.

Rule 1 – Packages: A package is formed by a set of interrelated classes. Therefore, a database is defined for each package in the conceptual schema that will store all the themes generated by the mapping related to it. The file name and its location are supplied by the user.

Rule 2 - Geographic Object Classes (◬): Each class mapped as Geographic Object generates at least one layer inside the corresponding database, whose spatial representation is defined according to the representation's stereotype. The attributes defined in the class are mapped as fields of a relational table.

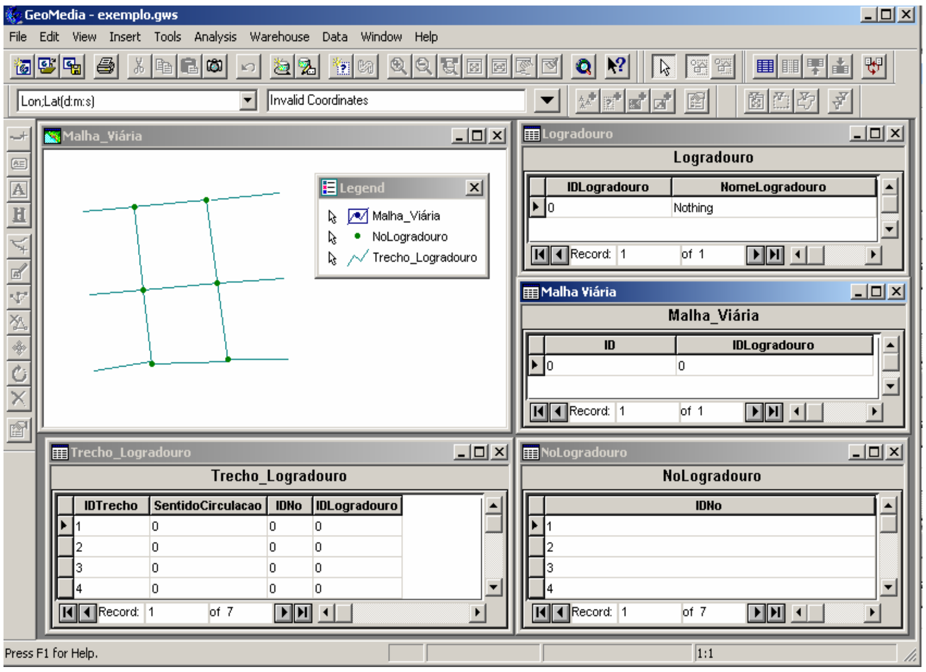
Rule 3 - Geographic Field Class (◬): The GeoMedia is a vector software, therefore, it is not possible to carry through an automatic mapping of the phenomena that are modeled as field. However, according to Goodchild [8], the geographic fields' representations are simply aggregations of points, lines and polygons, connected to spatial characteristics. A field with spatial representation of Isolines type, for example, can be mapped in a layer of line type, having a value associated to each line.

According to this analysis, the program considers some mapping options to the designer. Beyond the suggested attributes, the attributes of each class are also added to the table. If these approaches are not useful, the designer can choose not applying them and the geographic fields are mapped similarly to non-geographic objects.

Rule 4 - Non-Geographic Object Classes (△): Each class modeled as Non-geographic Object generates directly one relational table. Objects are classified as Non-geographic exactly for having no spatial representation.

Rule 5 - Relationships: Relationships as association, aggregation and composition are made in accordance to the specified multiplicity. There are basically three types of multiplicities: one-to-one (1..1); one-to-many (1..*); and many-to-many (*..*).This mapping follows the same rules used to generate relational DBMS, which are well known [5]. Relationship of generalization-specialization type can also be translated using the same solutions applied in relational DBMS. Indeed, the spatial representation must be considered accordingly.

As an example, the AGM-GeoMedia will create the logical-spatial schema shown in Figure 6 taking the data dictionary shown in Figure 4 as input.



Logradouro

IDLogradouro	NomeLogradouro
0	Nothing

Record: 1 of 1

Malha_Viária

ID	IDLogradouro
0	0

Record: 1 of 1

Trecho_Logradouro

IDTrecho	SentidoCirculacao	IDNo	IDLogradouro
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0

Record: 1 of 7

NoLogradouro

IDNo
1
2
3
4

Record: 1 of 7

Fig. 6. Data schema generated automatically for the GeoMedia

3.4. Analysis Patterns Manager Module

The idea of attaching an Analysis Patterns Catalog in a CASE tool is to help the database designer to find solutions that have already been used in similar GIS applications, which will improve the quality of the final database.

An extensive collection of analysis patterns, that permits search in the existing patterns and their use in a new project that is under development, are kept organized by this module.

The Analysis Patterns Catalog and its Manager compose the Analysis Patterns Manager Module. The Catalog Manager deals with the Analysis Patterns Catalog and keeps its file system organized. The Catalog is a set of analysis patterns where each analysis pattern is stored without dependence on another one. In fact, they are grouped in a directory system, according to the pattern's theme. Therefore, different patterns proposing solutions for a specific class of problems are stored in distinct files in the same directory.

Besides the schema supplied as solution by the Analysis Patterns, its documentation is also stored so that the reasoning behind a solution can be searched and analyzed. The analysis patterns' documentation is stored in a XML file sharing the same name of the file that has the pattern modeled. Both files are kept in the same directory in order to make search an easier task.

In the ArgoCASEGEO tool the designer can add new patterns in the directory structure. The designer itself defines the patterns' themes hierarchy. Usually, there isn’t an expressive number of analysis patterns available in an organization. Thus, designer groups can easily organize their patterns catalog in a simpler way. Analysis patterns can also be exchanged with partners from the same area. The tool also has import/export functions. Figure 7 shows an example of Analysis Patterns Directory.

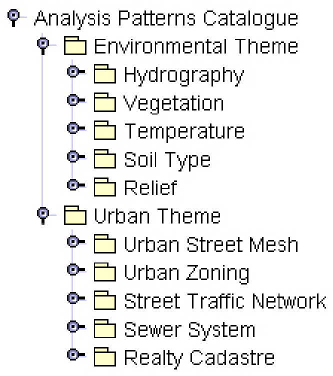


Fig. 7. An example of Analysis Patterns Directory

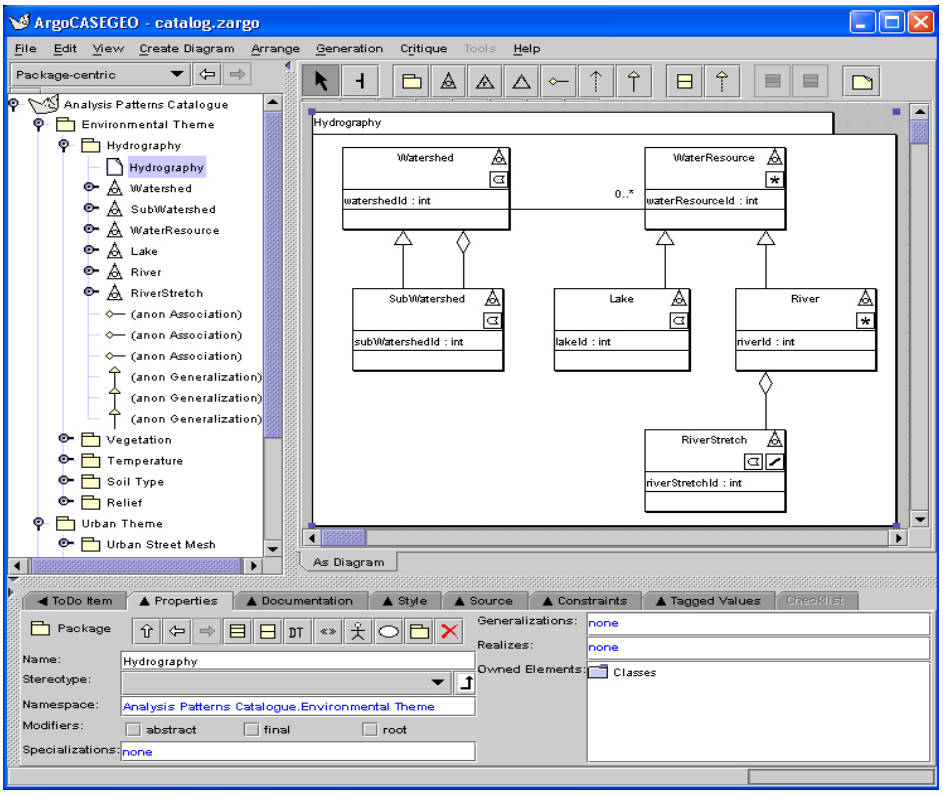
When the user needs to look for an analysis pattern, the manager builds a structure of packages, containing all the patterns recorded, in the Graphical Module. The Catalog Manager has a query mechanism which helps the designer to find analysis patterns by keywords that occur in the pattern’s documentation.

An example of the Catalog’s graphical environment can be seen in figure 8. On the left-hand side we can observe the directory hierarchy with the Hydrography pattern highlighted. All the pattern’s components are listed below the theme’s name. Opposite, the ArgoCASEGEO tool draws the schema related to the pattern. All the classes and relationships that form the schema can be identified and examined.

Putting the knowledge from the schema together with the information stored in the documentation we have enough means to understand and employ the pattern. As the documentation is stored in XML format, its recovering becomes easier and simpler to perform.

Figure 9 brings the Urban Street Mesh Pattern documentation source code. Every field found in the pattern template is kept in a special tag. As an example, the field Forces is represented by the pair of tags <forces> and </forces>. However, it might be interesting if we could store each force in a single pair of tags instead of keeping all the forces together. To solve this problem, a new pair of tag was created: <force> and </force>. Thus, to get the Forces data, it’s only necessary to go through the <force></force> tags.

All the fields from the pattern template are put in a more external pair of tags: <documentation> </documentation>. This group of information constitutes the Pattern’s documentation source code.



Name: Hydrography
Stereotype:
Namespace: Analysis Patterns Catalogue.Environmental Theme
Modifiers: abstract final root
Specializations: none
Generalizations: none
Realizes: none
Owned Elements: Classes

Fig. 8. The Analysis Patterns Catalog’s graphical environment representing the analysis pattern Hydrography

4. Conclusion

The CASE tool use during the GIS applications development makes creation time smaller, which, consequently, reduces cost. Moreover, the geographic databases quality increases.

The ArgoCASEGEO tool was implemented to assist the designer to develop its GIS applications with higher quality, following a design methodology based on a conceptual model specific for geographic databases and on a reusable collection of analysis patterns. The documentation produced during the project (e.g.: conceptual schema and data dictionary) permits further references and visualization, which makes future system maintenance easier and the immediate generation of new versions of the application with the updates. The data dictionary storage in XML/XMI format allows the schema exchange and can be used by other applications, for example, analysis patterns discovery and search automatic tools. The Analysis Pattern Manager Module organizes all the patterns recorded into a directory architecture, which raises the efficiency while searching for a new analysis pattern to be used.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<documentation>
   <problem>Which elements belong to a city’s street mesh?
   </problem>
   <context>Every city in Brazil (and probably in the world)
           has shown the same organization pattern, which is
           structured by their pathways organization. The set
           of pathways stretches generates an urban street
           network.
   </context>
   <forces>
      <force>Each drive way stretch is considered a road
           instance and should have an identification code
           and a name. It normally should be divided into
           several segments as well.
      </force>
      <force>A road stretch is a pathway segment between two
           connections.
      </force>
      <force>The set formed by the connections (or terminal
           points) and road stretches create an urban street
           mesh.
      </force>
   </forces>
   <participants>The StreetMesh class is a geographic
           phenomenon represented by a complex spatial object
           (represented by the u symbol). In this class many
           attributes may be defined relating to the network
           as a whole. Road is a conventional class
           implemented normally as a table in a relational
           DBMS. Each road is made of several road stretches,
           which corresponds to a network arc. A road stretch
           may be connected to other stretches but this
           connection is represented by the Crossroad class’
           instances, which are the network nodes. The
           network elements’ manipulation operations may be
           implemented as classes’ methods from StreetMesh,
           RoadStretch and Crossroad depending on their
           functionality.
   </participants>
   <related_patterns>The Urban Street Mesh uses the “State
           Across a Collection” pattern when modeling the
           Road and Road Stretch phenomena. Moreover, a new
           pattern project may be abstracted to create any
           network structure model made by nodes and arcs,
           whose topology’s relationship among its elements
           is kept to make possible common network operations
           such as the shortest path calculation (a value for
           each is necessary), network navigation, distance
           between nodes, etc.
   </related_patterns>
</documentation>
```

Fig. 9. The Urban Street Mesh Pattern documentation source code

The Reverse Engineering Module development, the implementation of a new AGM for the OpenGIS feature model, and new analysis patterns mining and specification in different domains are futures works.

Acknowledgements

This work has been partially supported by CNPq (Brazilian National Research Council), the Brazilian governmental agency for scientific and technological development.

References

1. Bédard, Y.: Visual modelling of spatial databases towards spatial extensions and UML. Geomatica, v.53, n.2, (1999).
2. Borges, K. A. V.; Davis Jr, C. D. Laender, A.H.F.: OMT-G: an object-oriented data model for geographic applcations. GeoInformatica, v.5, n.3 (2001).
3. Chrisman, N.: Exploring Geographic Information Systems. John Wiley & Sons (1997).
4. Coad, P.: Object Models: Strategies, Patterns, and Applications. 2nd ed. New Jersey, Yourdon Press (1997).
5. Elmasri, R.; Navathe, S. B.: Fundamentals of Database Systems. Addison-Wesley (2000).
6. Fernandez, E. B.; Yuan, X.: An analysis pattern for reservation and use of reusable entities. Procs. of Workshop in the Conference of Pattern Language of Programs – Plop (1999).
7. Fowler, M.: Analysis Patterns: reusable object models. Menlo Park, Addison Wesley Longman (1997).
8. Goodchild, M. F.: Geographical data modelling, Computers & Geosciences, v.18, n.4 (1992).
9. Hay, D. C.: Data Model Patterns: conventions of thought. New York, Dorset House Publishing (1995).
10. Isoware: CASE-Toll REGIS. (2002). Available in http://www.isoware.de/.
11. Johannesson, P.; Wohed, P.: The deontic pattern – a framework for domain analysis in information systems design. Data & Knowledge Engineering, v.31 (1999).
12. Kösters, G. et al.: GIS-Application Development with GeoOOA. Int. Journ. GIS, v.11, n.4 (1997).
13. Lbath A., Pinet, F.: The Development and Customization of GIS-Based Applications and Web-Based GIS Applications with the CASE Tool AIGLE. In Proc. 8th ACM GIS (2000).
14. Lisboa Filho, J.; Iochpe, C.: Specifying analysis patterns for geographic databases on the basis of a conceptual framework. In Proc.7th ACM GIS, Kansas City (1999).
15. Lisboa Filho, J.; Iochpe, C.; Borges, K. A. V.: Analysis Patterns for GIS Data Schema Reuse on Urban Management Applications. CLEI Electronic Journal v.5, n.2 (2002).
16. Lisboa Filho, J.; Iochpe, C.; Beard, K.: Applying Analysis Patterns in the GIS Domain. In Proc. 10th Annual Colloquium of the SIRC, Dunedin, NZ (1998).
17. Lisboa Filho, J.; Pereira, M. A.: Desenvolvimento de uma ferramenta CASE para o Modelo UML-Geoframe com Suporte para Padrões de Análise. In the proceedings of the IV Simpósio Brasileiro de Geoinformática – GEOINFO`02, Caxambu (2002). (in Portuguese)
18. Object Management Group: Meta Objects Facility (MOF) Specification. (2000).
19. Parent, C. et al.: Spatio-temporal conceptual models: data structures + space + time. In Proc.7th ACM GIS, Kansas City (1999).
20. Wohed, P.: Tool support for reuse of analysis patterns – a case study. In: A. H. F. Laender, S. W. Liddle, V. C. Storey (eds): ER2000 Conference, LNCS 1920, 2000. Springer-Verlag Berlin Heidelberg (2000).