

Improving Productivity and Quality of GIS Databases Design using an Analysis Pattern Catalog

Victor de Freitas Sodré, Jugurta Lisboa F., Valério Moysés Vilela, and Marcus Vinícius Alvim Andrade

Department of Informatics
Federal University of Viçosa
Viçosa, MG, 36570-000, Brazil

{vsodre, jugurta, vvilela, marcus}@dpi.ufv.br

Abstract

An analysis pattern is any part of a requirement analysis specification that can be reused in the design of new information systems. Analysis patterns permit reuse in a higher level than object-oriented class specialization. This paper describes how an Analysis Patterns Catalog can be applied to improve the productivity during development of GIS application and, consequently, the quality of their databases. The catalog is attached to the ArgoCASEGEO, an open source CASE tool that supports the UML-GeoFrame conceptual model, specific for geographic databases design. The analysis patterns collection is stored in a database composing the Catalog, which is structured through directory architecture. Thus, searching for existing analysis patterns will be an easier and more efficient task.

Keywords: Analysis Patterns, CASE tool, Reuse, Geographic database design.

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 [20], 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 database conceptual modelling [6], [7], [9], [11] and [21]. 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.

Once the user has finished the conceptual modelling phase, he is ready to start the logical design, which consists of the transformation of the conceptual schema into a data schema compatible with the GIS data model that will be used. The stage of transforming a conceptual schema into a logical-spatial schema, and its settlement in a GIS software, can be made automatically by a CASE (Computer Aided Software Engineering) tool. Some of these previously mentioned conceptual models are supported by CASE tools: Perceptory [1], REGIS [10], AIGLE [13] and Publisher Java MADS [20].

This paper describes how an Analysis Patterns Catalog can be applied to improve the productivity of GIS application development and, consequently, the quality of their databases. The catalog is attached to the ArgoCASEGEO [16], an open source CASE tool for the UML-GeoFrame conceptual model [14], which is specific for geographic databases design. The conceptual schema elaborated by this tool is stored in XML (Extensible Markup Language) format, so can be easily accessed and used by other systems. This tool also provides an Automatic Generation Module, able to generate data schemas into formats usually found in commercial GIS. A special attention will be given to the Analysis Patterns Module that implements a Manager to the Analysis Patterns Catalog.

Section 2 presents the UML-GeoFrame Model. Section 3 describes what an analysis pattern is whereas section 4 describes the ArgoCASEGEO tool. The Analysis Patterns Manager Module is described in details in section 5. Finally, section 6 brings final considerations and future works.

2 The UML-GeoFrame Model

The conceptual modelling of a 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 [14].

Copyright (c) 2005, Australian Computer Society, Inc. This paper appeared at the Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005), University of Newcastle, Newcastle, Australia. Conferences in Research and Practice in Information Technology, Vol. 43. Sven Hartmann and Markus Stumptner, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

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 geographic information and the difference between conventional data and geographic objects/fields. These elements specification is made based on the stereotypes set shown in Figure 1.

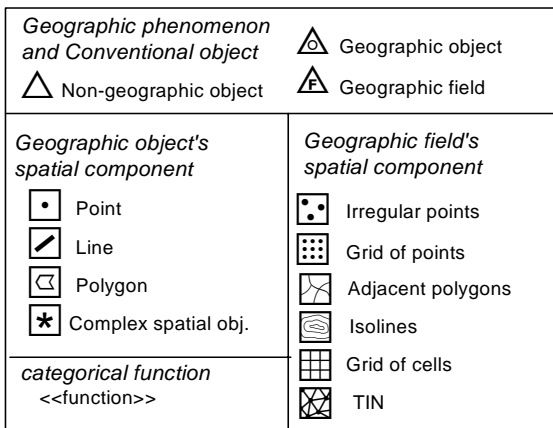


Figure 1: Stereotypes of the UML-GeoFrame Model

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 (\triangle) and Geographic Field (\triangle) classes, according to two

perception ways of the geographic phenomena, described by Goodchild [8]. Non-geographic Objects are conventional entities that are implemented as relational table. They are modelled on traditional form and are identified through the stereotype (\triangle).

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 combination of two or more stereotypes at 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 pair (\square \square).

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 illustrates a data schema extract using the UML-GeoFrame notation where three themes are seen (e.g.: COAL_ACTIV), modeled as UML packages. Each theme adjoins linked classes that may be subclasses of conventional object [\triangle] (e.g.: CoalCompany and LandUseType), subclasses of geographic phenomena perceived in object view [\triangle] (e.g.: City, Deposit) or geographic phenomena perceived in field view [\triangle] (e.g.: LandUseCover, Topography). Multiple representation examples are shown in CoalMine [\square \square], HydroResources [\square \square] and Topography [\square \square].

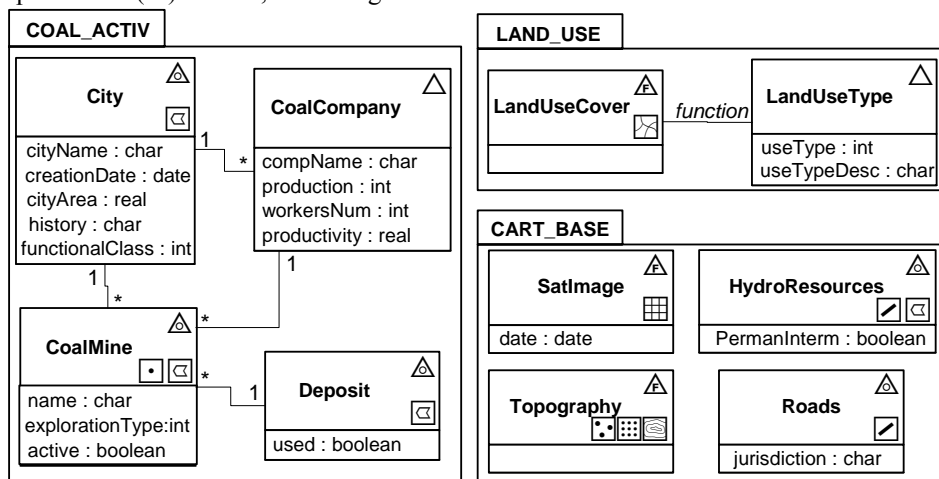


Figure 2: Stereotypes of the UML-GeoFrame Model

3 Analysis Patterns

An analysis pattern is any requirement specification part that begins in a project and may be reused by others [7]. The analysis pattern describes a set of real world objects, its relationships and rules that define its behavior and states in arbitrary level of abstraction [11].

Johannesson [11] identifies two quality requirements that a pattern must have. Firstly, it must be generic enough for example an analysis pattern must be applicable in different systems. Secondly, it must be easier to

understand (and use) than modelling a new application from the very beginning. Certainly these two requirements have conflicts once highly generic patterns may be hard to fit into new projects. That is a great challenge to patterns documenting designers: to find not only the balance between describing an abstract pattern in such a way it can be reused, but also that they can be easily understandable for future users.

The discovering or capturing process of a new analysis pattern requires enough analyst's skills and experience to realize existing analogies inside design parts of a studied

system that may be happening in other systems. According to Fernandez [6] the patterns creating process is made through design parts identification that are potentially reuse candidates. After that this sub-diagram must be generalized. Its details must be eliminated and the pattern must be documented according to a chosen pattern description language.

An Analysis Patterns should be described by these following fields (template): Problem, Context, Forces and Solution. Moreover, a pattern can include other optional fields such as an use example, participants, and related patterns [18].

The Problem field supplies a brief declaration of the problem that needs to be solved. The Context describes the scenery in which the problem was identified and its presented solution. The set of restrictions considered when solving the problem will be found in the Forces field. Finally, the Solution field supplies a class diagram with the proposed solution.

To exemplify the use of analysis patterns in the GIS area, one analysis pattern is presented below. It was identified from several database conceptual model analyses in urban applications [15]. Only essential operations and attributes in the given example are shown to shorten this work.

Problem: Which elements belong to a city's street mesh?

Context: Every city in Brazil (and probably in the world) has shown the same organization pattern, which is structured by their pathways organization (e.g.: streets, avenues, drives). The set of pathways stretches generates an urban street network.

Forces: 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. A road stretch is a pathway segment between two connections. The set formed by the connections (or terminal points) and road stretches create an urban street mesh.

Solution: Figure 3 shows the classes forming the pattern diagram.

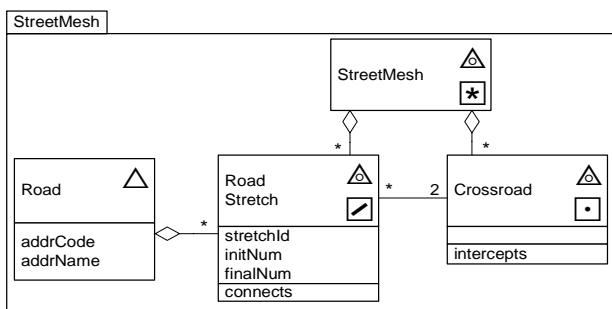


Figure 3: Urban Street Mesh pattern classes diagram

Each geographic phenomenon's pattern specifies the most general properties (attributes and operations), which must be extended and specialized for their specific application.

Participants: The StreetMesh class is a geographic phenomenon represented by a complex spatial object (⊠). 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 composed 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 manipulating operations of the network elements may be implemented as class methods from StreetMesh, RoadStretch and Crossroad depending on their functionality.

Related Patterns: The Urban Street Mesh uses the "State Across a Collection" pattern [4] 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 topological relationship among its elements is kept to enable common network operations such as the shortest path calculation, network navigation, distance between nodes, etc.

4 The ArgoCASEGEO Tool

The ArgoCASEGEO [16] 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 [19], a syntax for conceptual schema storage in XML documents [19].

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 4 illustrates the five-module architecture of the ArgoCASEGEO tool.

The Graphical Module allows the design of conceptual schemas, providing a set of constructors from 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 Reverse Engineering Module, not yet implemented, will enable the designer to get conceptual schemas from existing GIS applications. And finally, the Analysis Patterns Catalog and its manager are defined in the Analysis Patterns Manager Module. The following sections describe these modules giving further details.

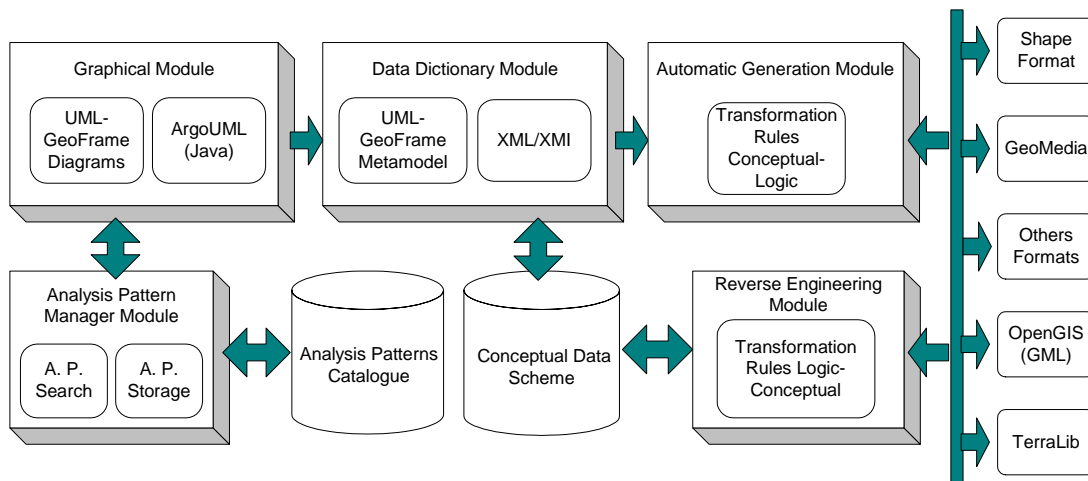


Figure 4: The ArgoCASEGEO Tool Architecture

4.1 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 5 illustrates the ArgoCASEGEO environment.

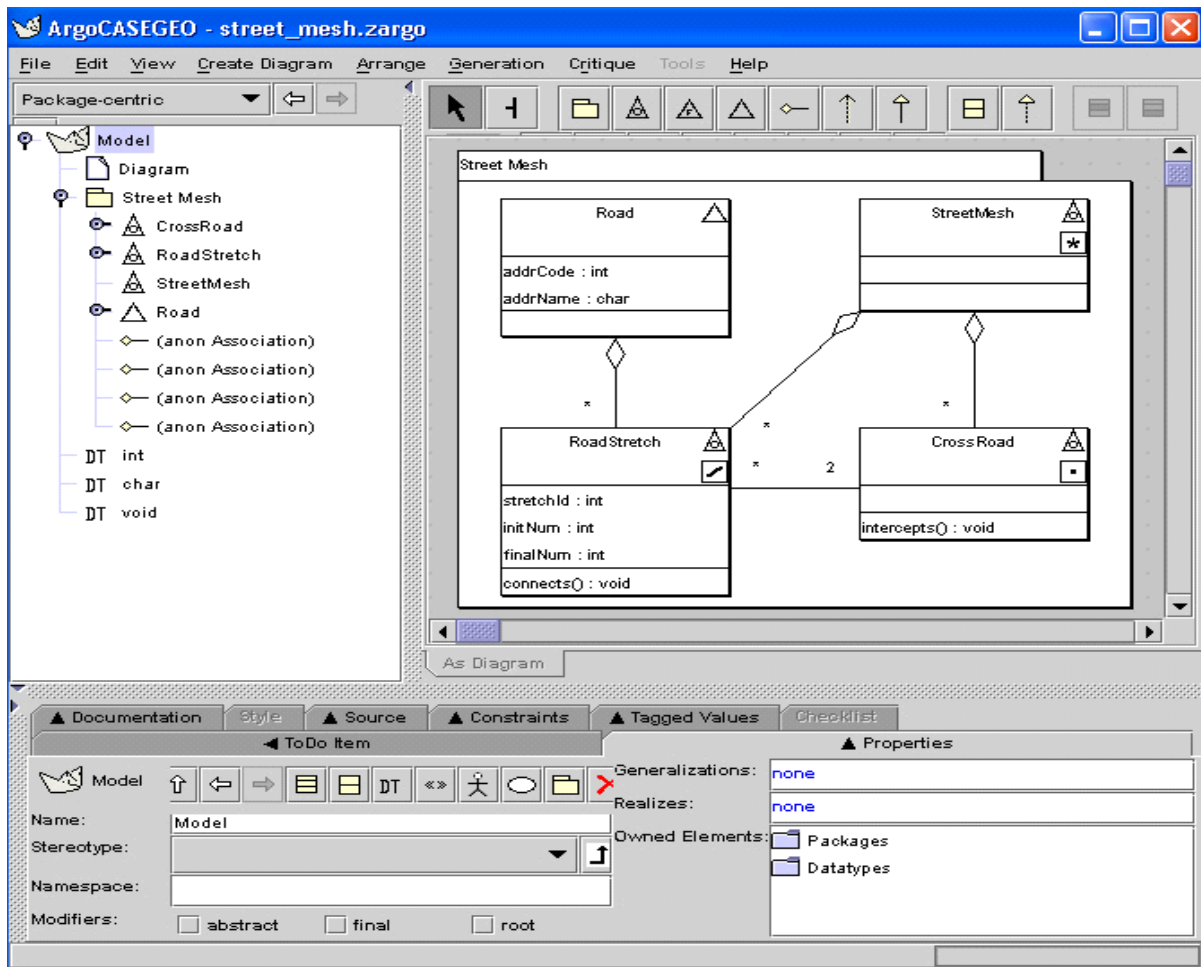


Figure 5: The ArgoCASEGEO graphical environment representing the analysis pattern Urban Street Mesh in the UML-GeoFrame model

An UML-GeoFrame data schema can be saved as a new Analysis Pattern, which can be (re) used in new data schemas, 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.

4.2 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. A tag that contains the class name, its spatial representations and its features delimits every class. The feature tag has two sub levels corresponding to the attributes and operations storage.

Figure 6 exemplifies the data dictionary of the Road Stretch class (modelled in figure 5) whose spatial representation is line type and its attributes are stretchId, initNum and finalNum. 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>RoadStretch
  </Foundation.Core.ModelElement.name>
  <Foundation.Core.GeneralizableElement.isLine xmi.value="true"/>
  <Foundation.Core.Classifier.feature>
    <Foundation.Core.Attribute>
      <Foundation.Core.ModelElement.name>stretchId
    </Foundation.Core.ModelElement.name>
    <Foundation.Core.Classifier xmi.idref="xmi.16"/>
  </Foundation.Core.Attribute>
  <Foundation.Core.Attribute>
    <Foundation.Core.ModelElement.name>initNum
  </Foundation.Core.ModelElement.name>
  <Foundation.Core.Classifier xmi.idref="xmi.14"/>
  </Foundation.Core.Attribute>
  <Foundation.Core.Attribute>
    <Foundation.Core.ModelElement.name>finalNum
  </Foundation.Core.ModelElement.name>
  <Foundation.Core.Classifier xmi.idref="xmi.12"/>
  </Foundation.Core.Attribute>
  </Foundation.Core.Classifier.feature>
</Foundation.Core.GeographicObject>

```

Figure 6: 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 classes' references that participate in the relationship, defines 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.

4.3 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. So, the ArgoCASEGEO tool needs a specific Automatic Generation Module (AGM) for each GIS. Two AGM have already been implemented in the ArgoCASEGEO. The first one transforms UML-GeoFrame schema into Shape format, used in the ArcView 3.2 (ESRI) GIS [17]. The second AGM, described in [16], transforms

conceptual UML-GeoFrame schema into logical-spatial schema for the GeoMedia GIS.

The major focus of this paper is on the Analysis Patterns Manager Module. Next section presents it in depth details.

5 The Analysis Patterns Manager Module

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

The Analysis Patterns Catalog and its Manager compose the Analysis Patterns Manager Module. Further details of both structures will be given below. 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.

An extensive collection of Analysis Patterns that permits search in existing patterns and their use in a new project that is under development are kept organized by this module. Before detailing the Analysis Patterns Manager Module, next section describes how an analysis patterns can be specified giving one example from the Urban domain.

5.1 The Analysis Patterns Catalog Manager

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 the others. In fact, they are grouped in a directory system according to the pattern theme. Therefore, different patterns proposing solutions for a specific class of problems are stored in distinct files in the same directory.

In the diagram, each catalog directory is represented as a package containing all the classes and diagrams of the stored pattern involved.

The analysis patterns' documentation is stored in a XML file sharing the same file name of 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 simple way. Analysis patterns can also be exchanged among peers. Also, the tool has import/export functions. An example of Analysis Patterns Directory is showed in figure 7.

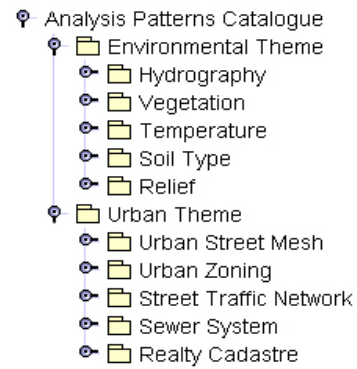


Figure 7: An Analysis Patterns Directory example

When the user needs to look for an analysis pattern, the manager builds in the Graphical Module a structure of packages containing all the patterns recorded. 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 Hydrograph 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.

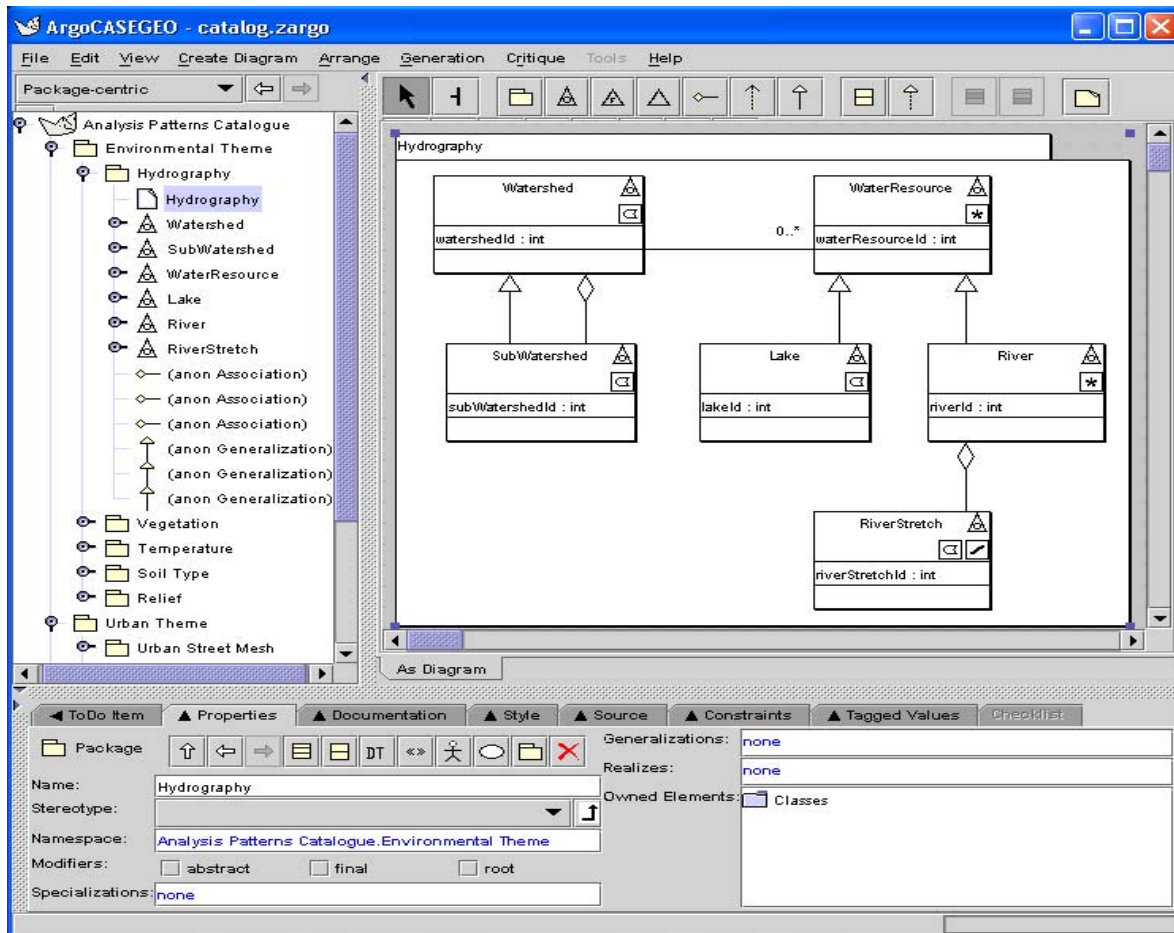


Figure 8: The Analysis Patterns Catalog's graphical environment representing the analysis pattern Hydrography

Putting the knowledge from the schema together with the information stored in the documentation we have enough resources 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 forms the Pattern's documentation source code.

```
<?xml version="1.0" encoding="UTF-8"?>
<documentation>
  <problem>Which elements belong to a city's street mesh?</problem>
  <context>Almost every city in the world has shown the same organization pattern, which is
    structured by their pathways organization (e.g.: streets, avenues, drives). 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 identifica-
      tion 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 ☒ symbol). 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, network navigation, distance
    between nodes, etc.
  </related_patterns>
</documentation>
```

Figure 9: The Urban Street Mesh Pattern documentation source code

6 Conclusions

The Analysis Pattern Manager Module for the ArgoCASEGEO tool organizes all the patterns recorded into a directory architecture, which raises the efficiency while searching for a new analysis pattern to be used. Thereby, time employed during the conceptual phase will be lessened and, consequently, cost reduction will be verified. Also, geographic databases quality increases following this design methodology based on a reusable collection of analysis patterns.

An analysis pattern does not need to present a sophisticated solution. Patterns should document tested and validated solutions, such as recurring problem's solutions. Unique problem's solutions do not need to be documented as a pattern because they probably will not be required again. Thus, the evolution of pattern documentation should be done naturally by other designers. Critical contributions and comments are always welcome in a pattern.

However, for this approach's success a cooperation culture among system developers should be cultivated. Instead of using patterns without putting their own data

available, users could work together and broadcast better patterns.

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. The analysis patterns stored in the Catalog are not complete solutions as seen above. Patterns describe advices, initial designs and usual problem's solutions. Analysis patterns must be adjusted for each case. We could also point, as future works, experiments to measure the improvement in productivity and quality provided by this case tool.

Acknowledgements

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

7 References

- [1] Bédard, Y. (1999): *Visual modelling of spatial databases towards spatial extensions and UML*. Geomatica, v.53, n.2.

- [2] Borges, K. A. V., Davis Jr, C. D. Laender, A.H.F. (2001): *OMT-G: an object-oriented data model for Geographic Applications*. *GeoInformatica*, v.5, n.3.
- [3] Chrisman, N. (1997): *Exploring Geographic Information Systems*. John Wiley & Sons.
- [4] Coad, P. (1997): *Object Models: Strategies, Patterns and Applications*. 2.ed. New Jersey: Yourdon Press.
- [5] Elmasri, R., Navathe, S. B. (2000): *Fundamentals of Database Systems*. 3.ed. Addison-Wesley.
- [6] Fernandez, E. B., Yuan, X. (1999): An analysis pattern for reservation and use of reusable entities. *Procs. of the Conference of Pattern Language of Programs*.
- [7] Fowler, M. (1997): *Analysis Patterns: reusable object models*. Addison Wesley Longman, CA.
- [8] Goodchild, M. F. (1992): *Geographical data modelling*. *Computers & Geosciences*, v.18, n. 4.
- [9] Hay, D. C. (1995): *Data Model Patterns: conventions of thought*. New York, Dorset House Publishing.
- [10] Isoware. (2002): *CASE-Toll REGIS*.
- [11] Johannesson, P.; Wohed, P. (1999): *The deontic pattern – a framework for domain analysis in information systems design*, *Data & Knowledge Engineering*, Vol.31.
- [12] Kösters, G. et al. (1997): GIS-Application Development with GeoOOA. *Int. Journ. GIS*, v.11, n.4.
- [13] Lbath A., Pinet, F. (2000): The Development and Customization of GIS-Based Applications and Web-Based GIS Applications with the CASE Tool AIGLE. *In Proc. 8th ACM GIS*.
- [14] Lisboa Filho, J.; Iochpe, C. (2000): Specifying analysis patterns for geographic databases on the basis of a conceptual framework. *In Procs 7th ACM GIS*.
- [15] Lisboa Filho, J.; Iochpe, C.; Borges, K. A. V. (2002): Analysis Patterns for GIS Data Schema Reuse on Urban Management Applications. *CLEI Electronic Journal*.
- [16] Lisboa F., J.; Sodré, V. F.; Daltio, J.; Rodrigues Jr., M. F.; Vilela, V. M.. (2004): A CASE tool for geographic database design supporting analysis patterns. *In Proc. of Conceptual Modeling for Advanced Application Domains. 1st Int. Workshop on Conceptual Modelling for GIS (CoMoGIS – ER2004)*, LNCS 3289, Springer Shanghai, China.
- [17] Lisboa Filho, J., Pereira, M. A. (2002): Desenvolvimento de uma ferramenta CASE para o Modelo UML-Geoframe com Suporte para Padrões de Análise. *In Proc. IV Simpósio GEOINFO*, Caxambu, Brasil. (In Portuguese)
- [18] Meszaros, G.; Doble, J. (1998): *A pattern language for pattern writing*.
- [19] Object Management Group. (2000): *Meta Objects Facility (MOF) Specification*.
- [20] Parent, C. et al. (1999): Spatio-temporal conceptual models: data structures + space + time. *In Proc.7th ACM GIS*, Kansas City.
- [21] Wohed, P. (2000): 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*.