Chapter 18

Developing Web Geographic Information System with the NDT Methodology

J. Ponce, A.H. Torres, M.J. Escalona, M. Mejías and F.J. Domínguez-Mayo

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/47971

1. Introduction

The increasing popularity of the Internet has led to the development of Web applications, known as Web GIS. A geographic information system (called GIS from now) is a software system that manages geo-referenced information. GIS systems are an automated system used for storing, analysing and manipulating geographical information. Geographical information represents objects and actions where geographical location is indispensable information (Aronof, 1989; Bull, 1994). In this context, a Web GIS system offers different GIS services for analysis and visualization of geographical information on the Web (Kim, 2002).

Thus, there is a growing need for tools and methodologies that support the rapid development of this kind of Web applications and their modification to meet the ever changing business needs. Recently, some frameworks have been developed Autodesk MapGuide, ESRI ArcIMS, gvSIG, Quantum GIS, PostgreSQL + PostGIS that enable users to develop and deploy Web GIS applications selecting functionalities a user needs.

In order to develop such systems, it is necessary to follow a proper development process. This process should cover the inherent characteristics of geographic information processing and also is available on Web platform. With this motivation, Escalona (Escalona et al, 2008) made a first approximation of such processes. They introduced a process for developing Web GIS (Geographic Information Systems) applications. This process integrated the NDT (Navigational Development Techniques) (Escalona & Aragon, 2008) approach with some of the Organizational Semiotic models (Liu, 2000). The use of the proposed development process is illustrated for a real application: the construction of the WebMaps system. WebMaps is a Web GIS system whose main goal is to support harvest planning in Brazil (Macário et al., 2007). The process can be seen on the next figure:

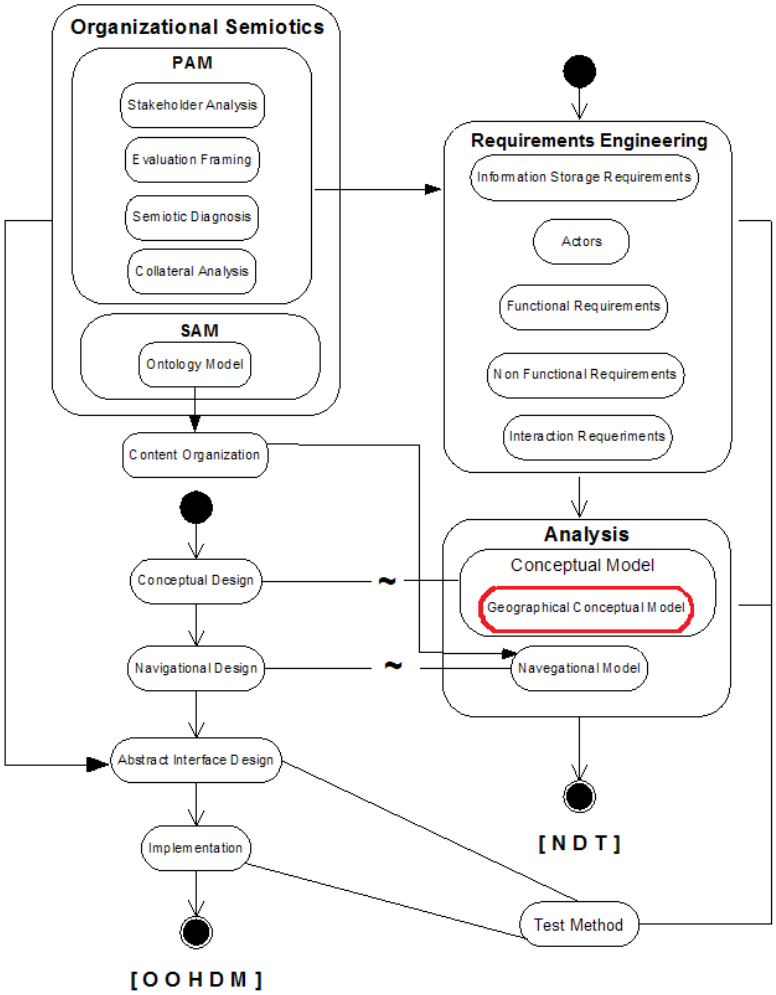

Figure 1. Development process for Web GIS applications (Escalona et al, 2008)

In the first activity, the process begins with a Requirement Engineering phase. Reinforce this phase with improving the process of capturing requirements GIS environments by defining a template for all those requirements. Next, an Analysis phase is performed. Later, the Abstract Interface model is built. Finally, the target system is implemented. Figure 1 shows the development process. Shows how models from NDT and from Organizational Semiotics are built following this process. The OOHDM (Oriented-Object Hypermedia Design Method) (Rossi, 1996) models are presented in the Figure 1 in order to reference the classic models. OOHDM is one of the most studied approaches. This approach for web developing is based on Hypermedia Design Model (HDM) (Garzotto et al., 1993) a structured approach to model hypermedia system.

The objective of the analysis phase is to detail, define and validate the requirements. In the NDT methodology, the analysis phase involves the definition of two main models: conceptual model and navigational model. A relational model might be generated from the conceptual model. The relational model is useful for the implementation of the databases of the system under development. NDT is helpful in the generation of navigational model for the pages that include information processing. However, NDT is not helpful for organizing static information. The Organizational Content approach is, then, used to complement the content and models generated by NDT. The Organization content used has been proposed and adapted from the ontology diagram by the research team at Institute of Computing Unicamp (Baranauskas et al., 2005). This process is left open and pending the definition and detailed specification Geographical Conceptual Model (highlighted in Figure 1). Therefore, the purpose of this paper is the formal definition of Conceptual Model inside the process for developing Web GIS systems.

This work is organized as follows. Section 2 introduces preliminary concepts needed to understand the process. Section 3 describes the formal definition of Geographic Conceptual Model inside development for Web GIS with NDT. Then, section 4 presents the case study based on formal definitions. Finally, section 5 outlines conclusions and future works.

2. Preliminary concepts

Then, will show some preliminary concepts. The first section will focus on Web GIS systems and the second; will outline briefly some concepts of NDT methods.

2.1. Web GIS

Web GIS are a special case of Web applications, meant to deal with complex geographic data and share them across several users for different business goals. Geographic information is usually distributed across different layers, which a Web GIS user should be able to handle separately or in overlay modality. Thus, besides common Web navigation and composition tasks, far more complex functionalities are needed in Web GIS for visualization and content management. As a matter of fact, with respect to traditional Web applications, this kind of systems require special focus on spatial data which may be acquired from different sources and stored in different formats, for all of which the user should be offered direct support. From a design point of view, Web GIS present many specific characteristics, making them different from traditional data intensive Web applications. Among them, two are of fundamental relevance: one is related to the data model and the other is concerned with the navigational model. As for the first aspect, the complex nature of geographic data, where the two components, descriptive and spatial, should be analyzed and managed in a joint manner requires the use of a different modelling approach, named Geodata and Metadata Conceptual Model. In our proposed methodology we will present the formal definition of this type of model with NDT. However, it is important to note the popularity of the E-R graphical notation, as well the intuitiveness of its extension to represent geographic data, led us to exploit the Spatial E-R

model as core of the Geodata and Metadata Modeling phase. According to this model, each set of geodata is described as a spatial entity characterized by a set of attributes, geometry and a couple of coordinates. Moreover, a similar extension has also been defined for relationships, so as to model entities which are spatially related, in terms of topological relationships. The expected output from this geodata design task consists of a set of logical schemas, which organize data in a relational way and associate them with layers, according to their meaning and the underlying data structure, which can be raster- or vector-model based. In addition, since a Web GIS application may also be aimed at supporting data sharing, metadata management is a crucial issue to be faced. Making interoperable data, which are heterogeneous in terms of formats and/or sources, may guarantee better performances during data exchanging and retrieving.

As for the navigational model a set of functionalities meant to dynamically navigate a map, commonly known as Web mapping. The generation of the Web Mapping model is not the aim of this work.

2.2. Navigational Development Techniques (NDT)

NDT (Navigational Development Techniques) is a model-driven Web methodology that was initially defined to deal with requirements on Web development. NDT starts with a goal-oriented phase of requirements and defines a set of transformations to generate analysis models.

NDT has evolved in the last years and offers a complete support for the whole life cycle. Figure x, represents processes supported by NDT. It covers viability study, requirements treatment, analysis, design, construction, implementation as well as maintenance and test phases as software development phases. Additionally, it supports a set of processes to bear out project management and quality assurance.

In the last year, it evolved to support different life cycles: sequential, iterative, agile processes, constitute some examples, although in Figure 2, the sequential life cycle was only shown to make the representation easier.

One advantage is that NDT can be used in the enterprise environment. Nowadays, a high number of companies in Spain work with NDT and the associated tools to develop software. This is possible due to the fact that NDT is completely supported by a set of free tools, grouped in NDT-Suite. This suite enables the definition and use of every process and task supported by NDT and offers relevant resources for quality assurance, management and metrics to develop software projects. NDT is based on the model-driven paradigm. It selects a set of metamodels for each development phase (requirements, analysis, design, implementation, construction, test and maintenance) in order to support each artifact defined in the methodology.

All concepts in every phase of NDT are metamodeled and formally related to other concepts by means of associations and/or OCL constraints. For instance, Figure 3, presents the metamodel for the requirements phase.

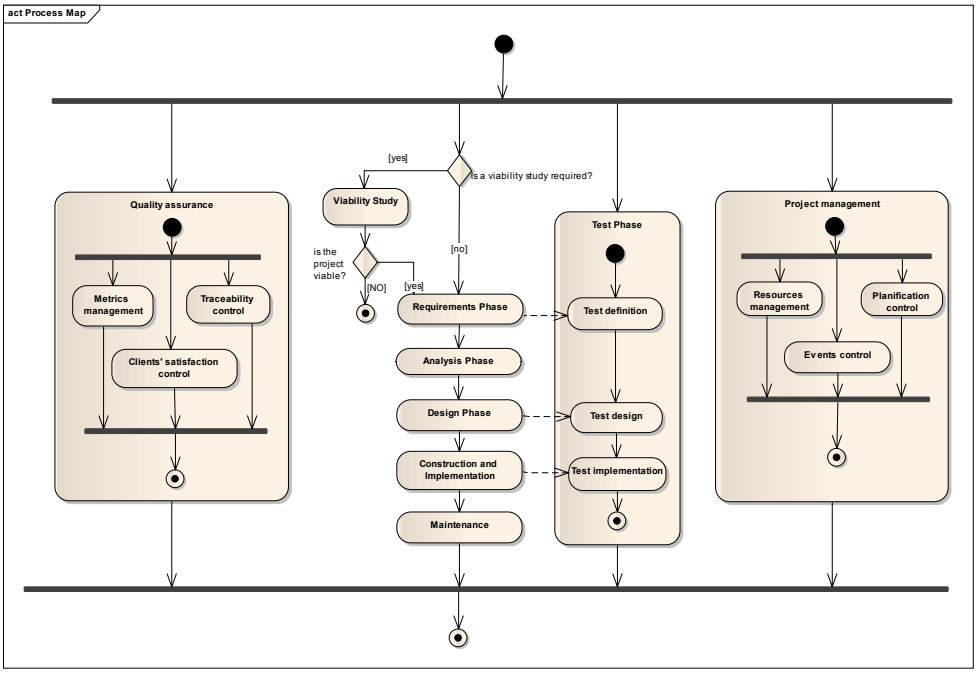

Figure 2. A general view of NDT Processes

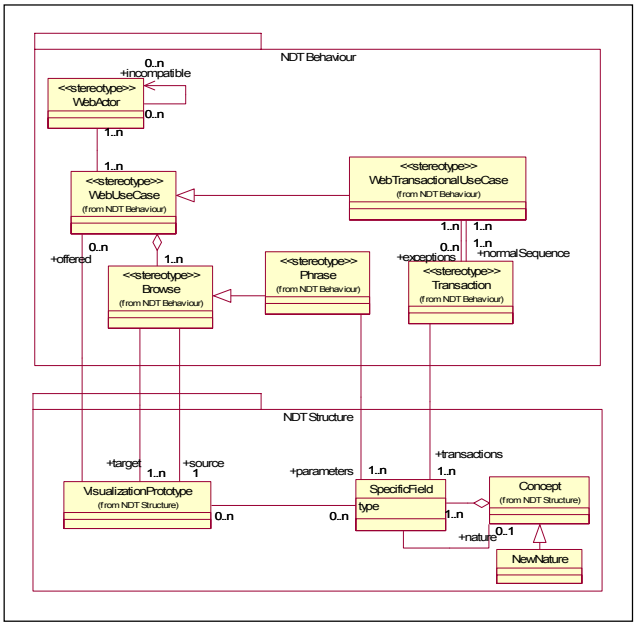

Figure 3. Requirements Metamodel in NDT

NDT proposes a set of QVT (Query/View/Transformation) among each metamodel for every phase, that enables to get one phase results from the previous one. Figure 4, shows this idea in a sequential development life cycle.

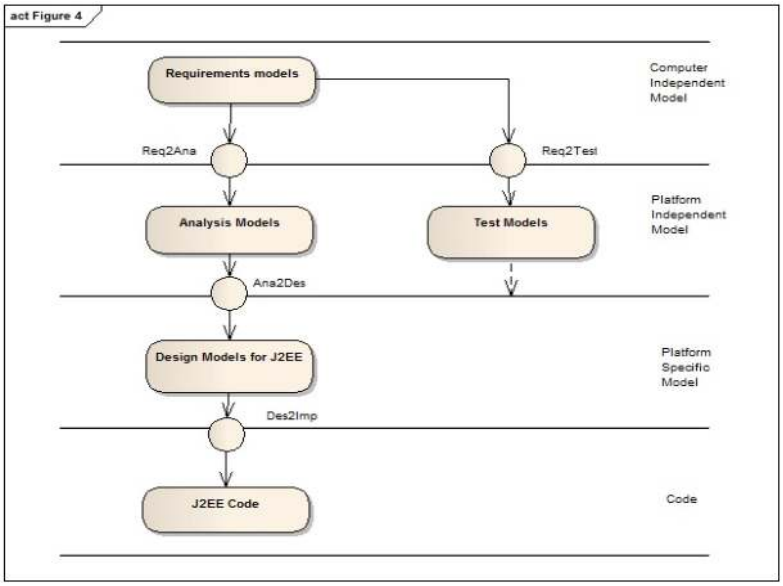

Figure 4. Transformations in a sequential life cycle

In this case, NDT defines transformations from the requirements model to the analysis model (Req2Ana), from the requirements model to Functional Test (Req2Test), from the analysis model to the design model (Ana2Des) and from the Design model to the implementation one (Des2Imp) is also available for Java code.

Nevertheless, using this framework in the enterprise environment is not possible. In companies, the use of metamodels, transformations and other elements is not real and technology seems too abstract for them.

For this reason, in 2004, we started two lines of work based on a new set of tools to support these NDT ideas:

1. Enterprise line: Offer an enterprise solution for the use of NDT.
2. Research line: Offer a way to validate transformations rules and metamodels.

After assessing different possibilities, some UML-profiles were developed for each NDT metamodel. These UML-profiles were defined in a UML-based tool named Enterprise Architect and then, the first tool for NDT-Suite was developed and called NDT-Profile.

To choose Enterprise Architect was not easy. In fact, a comparative study was carried out together by our research group and the Andalusian Government. The result was the tool that offered the best position in price/quality.

We selected UML profiles since, after some empirical studies, we realized that it was the easiest device for people in companies; on one hand, UML is commonly used in software companies, thus development teams already know its notation. On the other hand, they usually work with UML-based tools.

Other suitable possibility was using Eclipse and EMF technologies. However, as NDT is mainly oriented to requirements and end users' work, it does not offer as good interfaces as UML models, for instance, with use cases.

The problem of this election dealt with defining transformations. Enterprise Architect or other UML-based tools analyzed do not support QVT transformations. Thus, NDT transformations were translated into Java and implemented in a new tool that was included in NDT-Suite, named NDT-Driver.

This solution, NDT-Profile and NDT-Driver, provides development teams with a tool environment to define models as well as an engine to execute transformations.

These seeds prepare the environment for using NDT both for research and practical use.

Nowadays, NDT-Suite has been improved with some tools: NDT-Prototypes, NDT-Checker, NDT-Glossary, NDT-Report, NDT-Quality, NDT-Tracer and NDTQ-Framework. All of them bear out different aspects in NDT: quality assurance, exit generation or code checking, among others. Likewise, all of them are available on the Web.

This paper focuses on analyzing in depth tools that support, among other things, requirements validation and model-driven paradigm to reduce the development time and cost.

3. Developing web geographical information systems with NDT

The purpose of this section is to present the formal definition necessary to achieve the geographic conceptual model using NDT. This requires starting from the definitions of requirements engineering phase, then set the derivation to get the models of analysis. Therefore, the next section explains the relationship of the models in the existing methodology and is followed by extension for Web GIS.

3.1. Relationship models

NDT (Escalona et al., 2004) is a methodological process based on the navigation of web and hypermedia systems. NDT defines the Requirements Engineering and Analysis phases of a software development process. In the Requirements Engineering phase, NDT defines four models: information storage requirement model, functional requirements model, actors model, and interaction requirements model.

The information storage requirements model specifies the information needs of the system under development. This model answers the following questions: what information must the system store? What information does the system need?

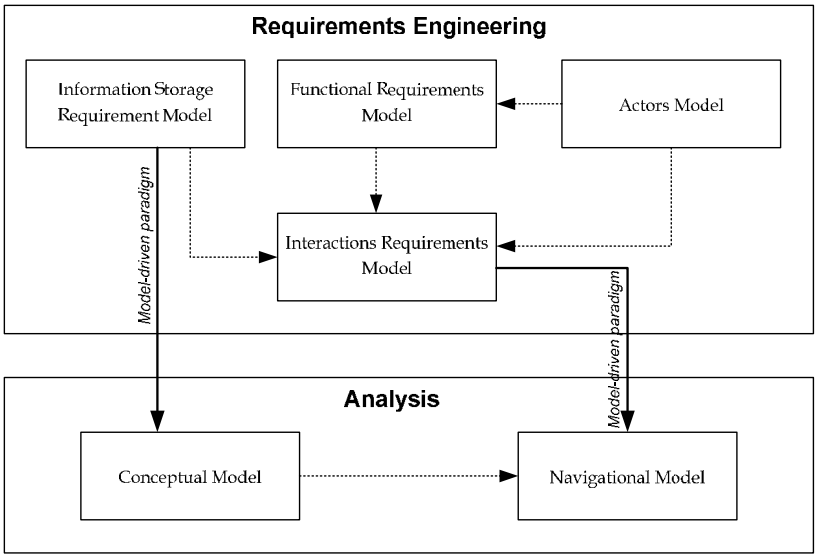

Figure 5. Relationship models

Functional requirements model specifies the operation of the system; this model answers the following question: what can the system do? The actors model specifies the roles of actors that interact with the system, their incompatibilities and generalization among them.

The interaction requirements model is a relevant model for a navigational system. The interaction requirement model provides the information and the functionality asked by the user.

In the Analysis phase, NDT defines two models. The conceptual model is a set of conceptual classes. These classes represent the static structure of the system. The navigational model is a class diagram with special classes that offers a view of the conceptual model and shows how to navigate through the information managed by the system.

The main documents obtained in NDT are: the System Requirement Document (SRD) in the Requirements Engineering phase and the System Analysis Document (SAD) in the Analysis phase.

3.2. Relationship models (extended to GIS)

Figure 6 shows the relationship extended for the treatment of Web GIS. In the phase of Requirements Engineering introduces the Geographical Data Storage Requirement Model, which is the basis for the derivation of Geographical Conceptual Model. The derivation of models is based on model-driven paradigm.

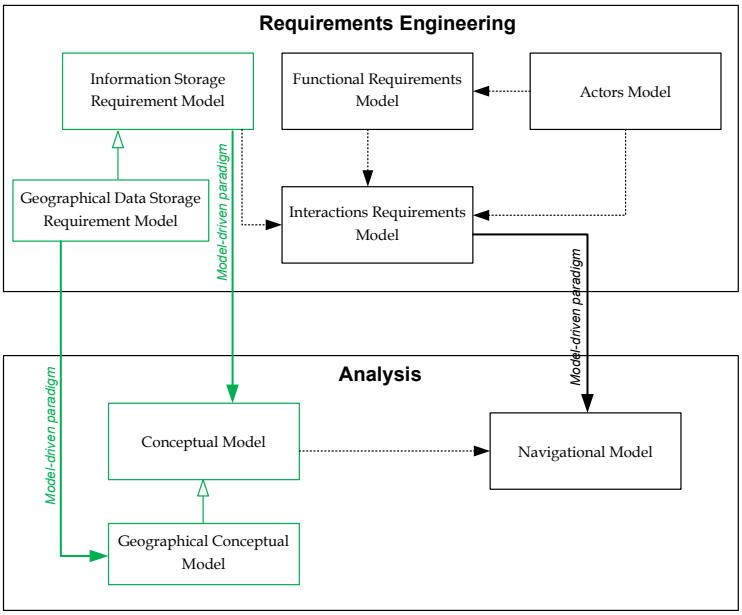

Figure 6. Relationship models (extended to GIS)

3.3. Geographical data storage requirement model

One of the keys for navigation in a navigational system is the information it handles. Some authors have defined the navigation system (Rossi, 1996)(Cachero, 2003) as a set of views of the information it stores and manages the system.

Being, therefore, the information displayed and handled one of the elements that are relevant for navigation during requirements engineering, is necessary to generate the model of storage requirements. This model represents the needs of storage that has the system and defines the characteristics of the information that will manage and display in the navigational system.

The structure of this model, including specification of geographic data, is presented in this section, by a corresponding metamodel, as well as more accurate description technique that is based on the use of patterns.

3.3.1. Model definition

The model of storage requirements contains a description of the information handled by the system and specifies its structure and meaning.

The items that appear in the model of storage requirements and the relationships are represented by the metamodel in Figure 7. We used the notation of the UML class diagram to describe it.

In this model, StorageRequirement class represents the set of information needs of the system. Instances of this class are used to define what information is handled in the system and its structure and meaning.

Each storage requirement is a relevant concept for the need to store information in the system. Each is described by an identifier, which turns out to be a unique code for every storage requirement, a name that represents it, and a description, which can define its meaning.

In addition, each storage requirement has associated a specific data set are represented in the diagram in Figure 7 by EspecificData class.

A specific piece of information is each of the specific items to be stored for a storage requirement. The storage requirement defines the general concept of information to manage the system, while the specific data concretely describes each of the items of information to be stored in a storage requirement.

The specific data are defined by a name that should be intuitive enough to describe the concept it represents, and a description detailing its meaning. In addition, each has specific data cardinality. The cardinality is a range that defines the minimum and maximum values of specific data that can be found in the requirement.

Another important concept is the nature of specific data. Nature defines the domain of that specific data. Let define the set of values and structural details that has the specific data. The concept of nature, although very close, does not match the data type concept. Nature represents a domain as a set of values that have a specific meaning within the system without going into details of low level, is the view that the user has over the domain and structure information.

Each nature, in turn, is defined by a short name so it must express its meaning. In the metamodel, it is possible to define three types of natures:

1. Predefined natures, which represent a set of predefined domains that are presupposed in any system and that the model of Figure 7 is represented by the class PredefinedNature. There are several predefined natures represented as child classes in the metamodel. Keep in mind that nature, as has been said, is not specific data type in the sense that is understood in the programming language, but not the broad sense of the programming language, since there is no indication that the type of this attribute is implemented later as a string, but in requirements specification, the user sees it as such. The final type of specific information or even the decision of whether this specific data is implemented as just a particular field is task of the designers and implementers of the system.
2. New natures, new domains that are defined in a concrete way for the system being modeled. In the class diagram in Figure 7 is represented by the class NewNature. Each new nature is described by an identifier, a name, which inherits from nature, and description, whose meanings are the same as for the storage requirements seen before. But it also has other attributes that do not have to take value in all cases, and they are:

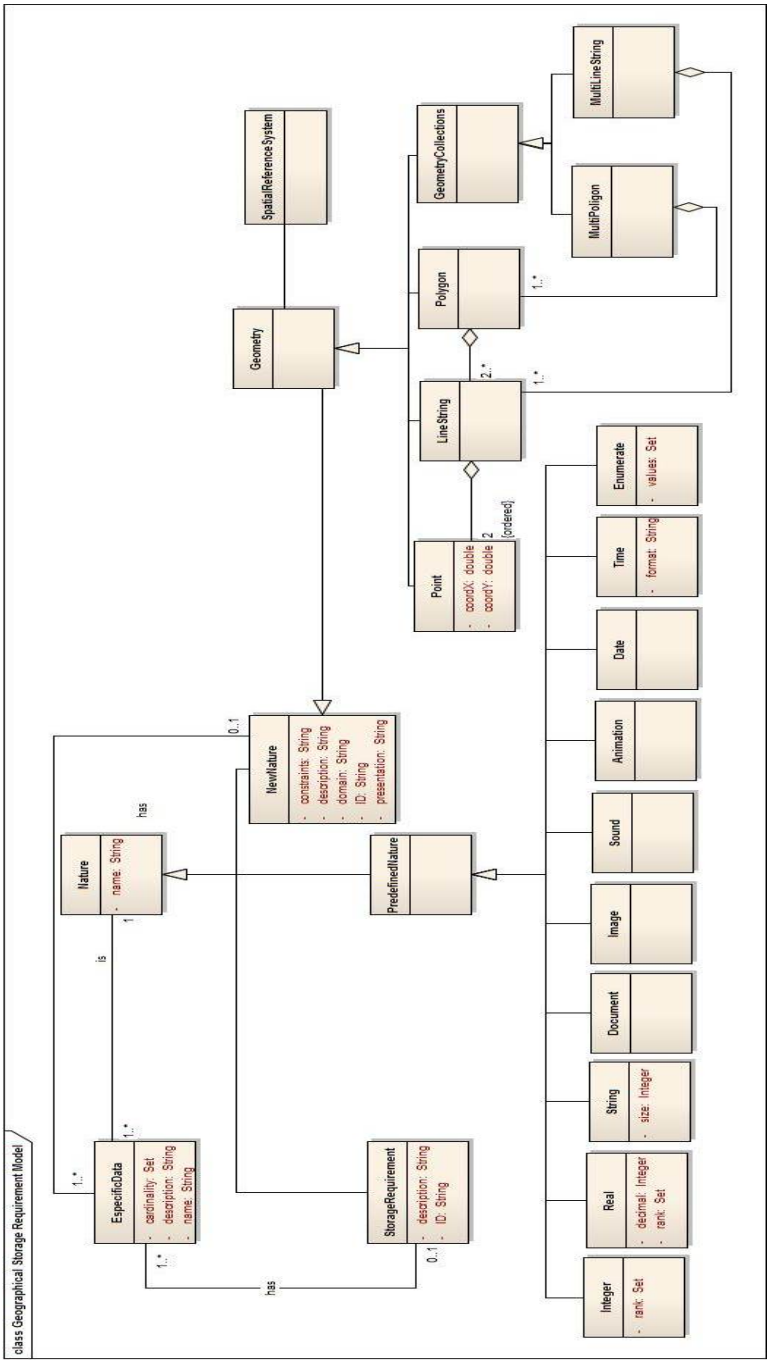

Figure 7. Geographical Data Storage Requirement Model

a. domain, representing the set of possible values that nature can take.
b. a set of constraints, which express constraints that must be fulfilled by the nature.
c. presentation restricts specific ways of how to represent.

Besides this, each new nature has an associated set of specific data whose meaning is similar to the storage requirements.

To own extension and adaptation to GIS requirements define the following elements of the previous figure.

- Geometry: is the root class of the hierarchy. Geometry is an abstract (non-instantiable) class. The instantiable subclasses of Geometry defined in this specification are restricted to 0, 1 and two dimensional geometric objects.
- SpatialReferenceSystem: identifies the coordinate system for all geometries, and gives meaning to the numeric coordinate values for any geometry instance. Examples of commonly used Spatial Reference Systems include 'Latitude Longitude', and 'UTM Zone 10'.
- Geometry Collection: is a geometry that is a collection of 1 or more geometries. All the elements in a GeometryCollection must be in the same Spatial Reference. This is also the Spatial Reference for the GeometryCollection. GeometryCollection places no other constraints on its elements. Subclasses of GeometryCollection may restrict membership based on dimension and may also place other constraints on the degree of spatial overlap between elements.
- Point: is a 0-dimensional geometry and represents a single location in coordinate space. A Point has a x-coordinate value and a y-coordinate value.
- LineString: is a curve with linear interpolation between points. Each consecutive pair of points defines a line segment.
- MultiLineString: is a collection whose elements are LineStrings.
- Polygon: is a planar Surface, defined by 1 exterior boundary and 0 or more interior boundaries. Each interior boundary defines a hole in the Polygon. Polygons are topologically closed.
- MultiPolygon: is a collection whose elements are Polygons

3. Storage requirement. Nature may be possible to turn a storage requirement. This means that the domain of this nature is represented by the set of elements that abstractly represents the storage requirement.

3.3.2. Model description

After specifying the elements involved in the model of storage requirements, we must study how to represent in the process. As mentioned, we propose the use of patterns as a technique of definition. Generically, for defining the elements of the storage requirements model proposes the use of two patterns.

The basic structure of the first, includes the aspects related to storage requirements that are described in the metamodel. This pattern, shown in Table 1 is an evolution of the proposed

staffing Environment Requirements Engineering Methodology for Information Systems (Durán, 1999).

It is assumed the structure of this proposal, but we have added new concepts such as cardinality or natures which are necessary to capture for the treatment of navigation as presented in the following chapters.

The meaning of the fields of this pattern is easily deduced from the definition of the metamodel.

The identifier is a code that identifies uniquely the requirement. Coding is proposed beginning with the letters RA, indicating that it is a requirement of storage, and continues for a unique sequence number. The use identifiers with a certain meaning and not, for example, numbers, is very convenient for quick identification of the requirement.

The descriptive name and description attributes match the name and description of the metamodel. As for the specific data field contains all the information regarding the specifics of the requirement. Each instance contains the attributes and relationships represented in the metamodel: the name and description, nature, and its cardinality and properties.

<ID>	<descriptive name of the requirement>	
Description	The system will store the information on the <concept relevant>. Specifically:	
Specifics data	**Name and description**	**Nature**
	<data name>:< brief description of the data>	<nature of the data> [Cardinality:<cardinality>]
	…	
	<data name>:< brief description of the data>	<nature of the data> [Cardinality:<cardinality>]

Table 1. Pattern for the definition of storage requirement

The second pattern is proposed to describe quite similar and the information concerning new natures. It is described in Table 2.

<ID>	<descriptive name of the nature that is being defined>	
Description	This nature represents <description of the nature>	
Specifics data	**Name and description**	**Nature**
	<data name>:< brief description of the data>	<nature of the data> [Cardinality:<cardinality>]
	…	
	<data name>:< brief description of the data>	<nature of the data> [Cardinality:<cardinality>]
Domain	<value domains of nature>	
Constraints	<constraints that need to have the fields of nature>	
Presentation	<description of the way it presents the fields of nature>	

Table 2. Pattern for the definition of new natures

3.4. Geographical conceptual model

The conceptual model represents the static structure of the system. Can model and describe the information handled by the system, and its structure. Classically been called static model, and although the concept is the same, in the surroundings of web engineering and navigational systems has extended the terminology of the conceptual model. It is in the web world of engineering where there has been more emphasis on the aspect of navigation, so it is assumed in this paper the terminology of the conceptual model.

Within the development of navigation, the conceptual model represents and describes the type of information will be treated, made or modified during the navigation process and within the navigational system and the relationships established between the items of this information . In this way, the navigation of a system is defined on the basis of the information handled.

3.4.1. Model definition

To describe the conceptual model is proposed using a class diagram for definition. As the authors of UML indicate, when using a class model to represent the static model of a system is used with one of these three objectives (Booch et al., 1999):

1. To model the vocabulary of a system, this involves making decisions about the limits of the system.
2. To model simple collaborations, or whatever it is, to represent the relationships and collaborations that are established in the society of classes, interfaces and other system elements.
3. To model a logical scheme of the database. The class model is the first model the system on the final design of the database.

The conceptual model is oriented mainly to the two first points. Allows to define the universe of discourse of the problem and the relationships that occur between the abstract elements defined in the model.

Figure 8 shows the metamodel that describes the conceptual model. This metamodel is based on the proposed by UML v.2.0. Note that this metamodel is a view of the UML metamodel proposed v.2.0. Actually, the class model includes many more aspects than those included in the metamodel of Figure 8. Concepts allowed in the class model as the invariant or stereotypes or other means of expansion are not included in our meta-model, but could be added in the conceptual model if desired. The metamodel in Figure 8 deals only with aspects of the relevant class model for defining the navigation system.

The main element of this model is the UML Classifier class v.2.0. Under this kind are shown the two main elements of the model classes: classes, included in Class, and partnerships, contained in the Association class.

All classifier is represented by an identifier, a name and description. These attributes (attributes) do not appear under this name in the UML metamodel, however, have remained in figure 8 for the metamodel classes follow the structure of previous models.

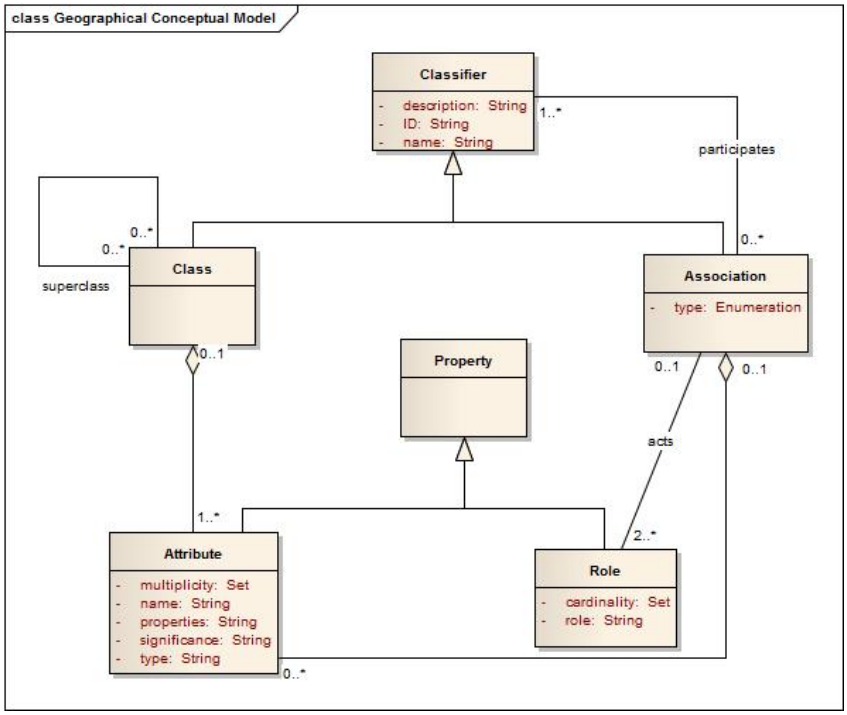

Figure 8. Conceptual model

Specifically entering the first class that inherits from the classifiers, the Class, say their class instances represent any conceptual system.

These may establish hierarchical structures that are represented by the association it has with herself. The school also has an associated set of attributes that describe them. These attributes are represented in the Attribute class, which according to the UML metamodel is a daughter v.2.0 Property class in UML v. 2.0.

With respect to Class Association, their instances represent the relationships can be established between different classifications of the model. Furthermore, in a partnership at least two roles, represented in the class Role, each described by a role name, contained in the attribute role, and a cardinality. These concepts are consistent with those proposed in UML and are not objective of this work to define them in detail.

3.4.2. Model description

To represent the conceptual class model proposes two techniques basically:

1. The class diagram, it is advisable to follow the standard UML notation.
2. The data dictionary. The data dictionary to describe the graphical model class diagram in a more concrete and detailed.

There are many possibilities to develop a data dictionary. This proposal recommends the use of patterns similar to those described for requirements engineering since, as already mentioned, offer a structured and simple description of the model and facilitates automation.

For the data dictionary conceptual class model, we propose two patterns. The first one is aimed to describe the system classes and is described in Table 3.

<ID>	<descriptive name of the class>	
Inherited from	<ID and name of the parent class>	
Description	The system will store the information on the <concept relevant>. Specifically:	
Attributes	**Description**	**Significance**
	name [multiplicity][:type][{Properties}]	<Meaning of attribute>
	…	
	name [multiplicity][:type][{Properties}]	<Meaning of attribute>

Table 3. Pattern for the definition of classes

This pattern is quite similar to those used so far and the meaning of its fields can easily guess from the metamodel. The attributes id, name and description are collected in fields that are labeled as his name. It is advised, as requirements engineering, follow specific criteria for assigning identifiers. In this case, it is advised that the class identifier begins with CL or CLn and will be followed by a unique number. By studying the criteria for referral in the next chapter, it indicates when you have to opt for a code that starts with CL or one that starts with CLn. The field 'inherits from', reflects the hierarchical relationships in which it participates as a child class and the attributes field gives the description of the attributes associated with the class.

The second proposed standard aims to provide a tool to describe the associations. The structure is displayed in Table 4.

<ID>	<descriptive name of the association>		
Description	Classes <identifiers of the classes that are related> are related by the association that represents <significance of the association>		
Type	{unidirectional, bidirectional}		
Classes	**Class name**	**Role name**	**Cardinality**
	<ID and name of the class>	<role of the association for the class>	<cardinality of the association in the class>
	…		
Attributes	**Description**		**Significance**
	name [multiplicity][:type][{Properties}]		<Meaning of the attribute>
	…		

Table 4. Pattern for the definition of associations

It's a fairly intuitive pattern also taking into account the above definitions. This pattern is intended only for associations, since the hierarchical relationships are reflected in the previous pattern. The pattern reflects associations in the fields ID, name, description and type attributes with the same name which appear in the metamodel. In the case of partnerships it is advisable to use an identified beginning with AS and will be followed by a unique sequence number.

Class field contains the reference to classes participating in the association keeping for each code and name of the class, the role it assumes in the association and the cardinality of the same. There is also the attributes field which gathers the attributes of the association in the event that you are defining what is a class association.

3.5. Basic conceptual model derivation

As mentioned, the basic conceptual model can be generated from the information provided by the elements of the model storage requirements engineering requirements.

There are different relationships between elements of the model storage requirements and the basic conceptual model. In Figure 9 are represented by a diagram of these classes. The classes participating in this diagram are classes that come from the metamodel presented in the previous chapter, specifically the metamodel storage requirements and conceptual metamodel. In the next figure has only had relationships that are reflected throughout this section and have ignored other metamodels shown in the previous chapter that are not relevant to the referral process.

Keep in mind that the added relationships, generates four labeled in the figure, only hold for the case of the basic conceptual model need not be met in the final model. In the figure, the thick line is divided for part of the diagram that comes from the storage requirements metamodel (which is at the top) and comes from the conceptual metamodel (which is at the bottom). The Objective of the analysis phase is to detail, define and validate the requirements. In the NDT Methodology, the analysis phase Involves the definition of two main models: conceptual model and navigational model. A relational model generated from Might Be the conceptual model. The relational model is Useful for the Implementation of the databases of the system under Development.

3.5.1. Derivation of classes from the storage requirements and natures

The first relationship can be studied in Figure 9 is what allows us to derive lessons from the natures and storage requirements. From it, one might conclude that, in the basic model, each storage requirement and every new nature generates a class that has the same name and same description or nature of the requirement that generates it.

In the previous section are advised to give IDs format requirements beginning with RA and natures starting with NA. It is also advisable to identify classes with identifiers that begin with CL and CLn.

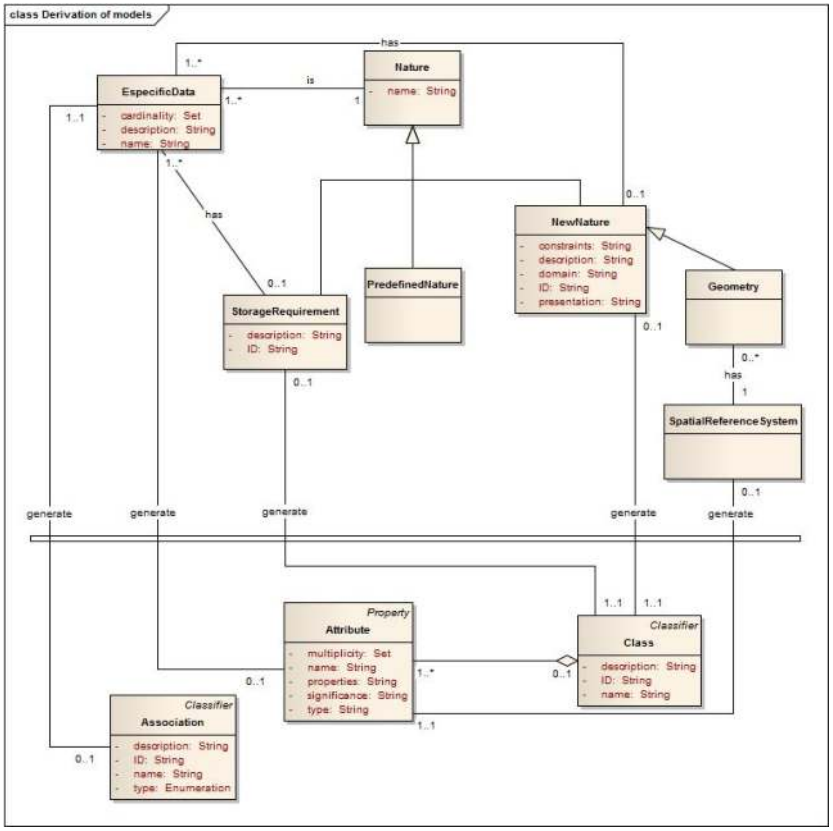

Figure 9. Relations between the model of storage requirements and conceptual model

3.5.2. Deriving attributes from the specific data

Another relationship that will allow to establish referral systems is that between specific data and attributes. Each class has an associated set of attributes, in the same way that every nature and every storage requirement is related to a specific data set.

Figure 9 shows a relationship which indicates that each specific data generated from 0 to 1 attribute. The specific data storage requirement or a new nature will serve to generate the attributes of the class that is generated from that specific data or from that nature. However, not all the specifics of the requirement or derived nature of the class attribute, taking into account only those whose nature is either a default or nature a new nature.

3.5.3. Derivation associations from specific data

The latest model of relationship between storage requirements and the basic conceptual model of Figure 9 to study is made between specific data and associations. This time, specific data are taken into account are those which are naturally a storage requirement.

Each of them will cause a binary association between two classes. So, is restricted to the beginning than the number of classifiers involved in the association are two and that its type is Class.

4. Case study

This will model the patterns described above, the requirements model and the conceptual model that includes the following case. Just try to store the geographic information world's rivers that pass through cities. The cities are in countries.

That is why we get information from both cities by passing a river crossing countries, etc. In each of the features shown in the example data is stored minimum and necessary.

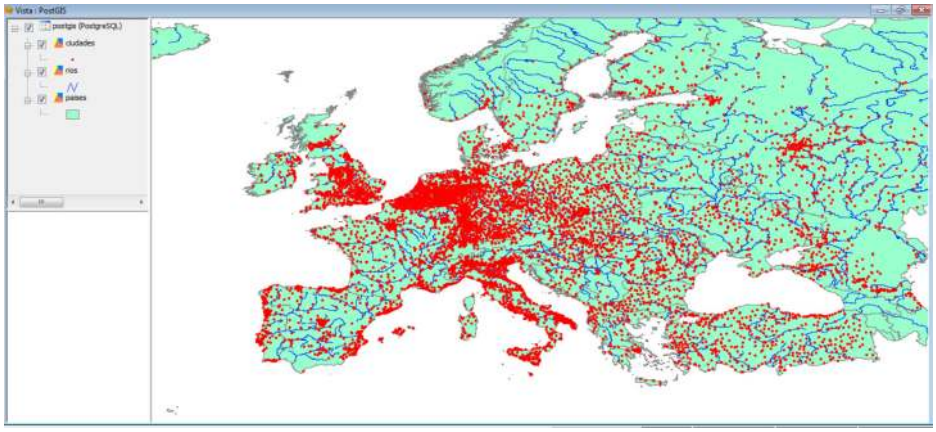

Figure 10. GIS Data Representation

Then define the patterns corresponding to the storage requirements.

RA-01	Country	
Objectives associated	OBJ-01: provide full view of geographic information layers	
Description	The system will store data for countries. Specifically:	
Especific Data	**Name and Description**	**Nature**
	codeCountry: stores information about the code that uniquely identifies each country	NA-01
	Name: stores information about the country's name	String
	Area: stores information about the country's area	Real
	Population: stores information about the country's population	Integer
	Capital: stores information about the country's capital	RA-02 Cardinality: 1..1
	layerCountry: layer stores the type which represents information	NA-04
	Cities: stores a list of cities	RA-02 Cardinality: 1..*

Table 5. Requirement pattern RA-01

RA-02	City	
Objectives associated	OBJ-01: provide full view of geographic information layers	
Description	The system will store data for cities. Specifically:	
Especific Data	**Name and Description**	**Nature**
	codeCity: stores information about the code that uniquely identifies each city	NA-02
	Name: stores information about the city's name	String
	Area: stores information about the city's area	Real
	Population: stores information about the city's population	Integer
	Country: stores the country that owns the city	RA-01
	layerCity: layer stores the type which represents information	NA-05
	Rivers: stores a list of rivers	RA-03 Cardinality: 0..*

Table 6. Requirement pattern RA-02

RA-03	River	
Objectives associated	OBJ-01: provide full view of geographic information layers	
Description	The system will store data for rivers. Specifically:	
Especific Data	**Name and Description**	**Nature**
	codeRiver: stores information about the code that uniquely identifies each river	NA-03
	Name: stores information about the river's name	String
	Length: stores information about the river's length. Be stored in kilometers	Real
	layerRiver: layer stores the type which represents information	NA-06
	Cities: stores a list of river crossing towns	RA-02 Cardinality: 1..*

Table 7. Requirement pattern RA-03

Then define the patterns corresponding to the natures.

NA-01	codeCountry	
Objectives associated	OBJ-01: provide full view of geographic information layers	
Description	This nature is the structure describing the unique code identifying each country.	
Especific Data	**Name and Description**	**Nature**
	country: store, through a code of three numbers, the country	String Size: 3
Presentation	Data are presented using a 3-digit numeric code.	

Table 8. Nature pattern NA-01

NA-02	codeCity	
Objectives associated	OBJ-01: provide full view of geographic information layers	
Description	This nature is the structure describing the unique code identifying each city within the country.	
Especific Data	Name and Description	Nature
	country: store, through a code of three numbers, the country	NA-01
	city: is a three-digit saves the particular number that has this city in the country.	String size: 3
Presentation	Data are presented using a 6-digit numeric code. The first three allude to the country and three from the city. Are separated by a point. Example: CCC.ccc	

Table 9. Nature pattern NA-02

NA-03	codeRiver	
Objectives associated	OBJ-01: provide full view of geographic information layers	
Description	This nature is the structure describing the unique code identifying each river.	
Especific Data	Name and Description	Nature
	river: store, through a code of six numbers, the river	String Size: 6
Presentation	Data are presented using a 6-digit numeric code.	

Table 10. Nature pattern NA-03

NA-04	layerCountry	
Objectives associated	OBJ-01: provide full view of geographic information layers	
Description	This nature is the structure describing the unique code identifying each country.	
Especific Data	Name and Description	Nature
	typeGeometric: defines the type of geometry of the layer	MultiPolygon
	dataGeometric:	NA-07
Presentation	Data are presented using a 3-digit numeric code.	

Table 11. Nature pattern NA-04

NA-05	layerCity	
Objectives associated	OBJ-01: provide full view of geographic information layers	
Description	This nature is the structure describing the unique code identifying each city.	
Especific Data	Name and Description	Nature
	typeGeometric: defines the type of geometry of the layer	Point
	dataGeometric:	NA-07
Presentation	Data are presented using a 3-digit numeric code.	

Table 12. Nature pattern NA-05

NA-06	layerRiver	
Objectives associated	OBJ-01: provide full view of geographic information layers	
Description	This nature is the structure describing the unique code identifying each river.	
Especific Data	**Name and Description**	**Nature**
	typeGeometric: defines the type of geometry of the layer	MultiLineString
	dataGeometric:	NA-07
Presentation	Data are presented using a 3-digit numeric code.	

Table 13. Nature pattern NA-06

NA-07	dataGeometric	
Objectives associated	OBJ-01: provide full view of geographic information layers	
Description	Defines the number of dimensions and spatial reference system	
Especific Data	**Name and Description**	**Nature**
	nDimensions	Integer
	spatialSystemReference	NA-08

Table 14. Nature pattern NA-07

NA-08	spatialSystemReference	
Objectives associated	OBJ-01: provide full view of geographic information layers	
Description	Define spatial reference system	
Especific Data	**Name and Description**	**Nature**
	code	Integer

Table 15. Nature pattern NA-08

5. Conclusion

This paper has presented the formal definition of geographic data models with NDT methods. It was based on the extension of the definition of existing models of complex navigation systems. He presented the definition of the model Engineering Requirement phase and the derivation of Analysis models. However, we are still working on the specifications of constraints of the models using OCL.

Finally, a Web GIS model is necessary to store information, which is discussed in this work, but it is also important to define the navigation-related models of this type of geo-referenced data. Figure 11 shows a future work, which is the generation of the Web Mapping Model.

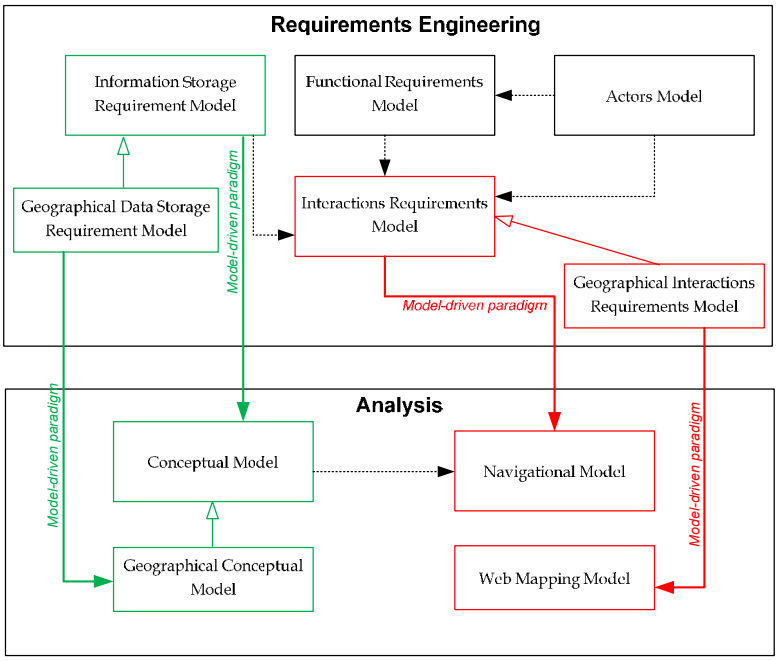

Figure 11. Generating Web Mapping Model (Future works)

Author details

J. Ponce, A.H. Torres, M.J. Escalona, M. Mejías and F.J. Domínguez-Mayo
Department of Computer Languages and Systems, University of Seville, Seville, Spain

Acknowledgement

This research has been supported by the project Tempros project (TIN2010-20057-C03-02) and Red CaSA (TIN 2010-12312-E) of the Ministerio de Ciencia e Innovación, Spain and NDTQ-Framework project of the Junta de Andalucia, Spain (TIC-5789).

6. References

Aronoff, S., 1989. Geographic Information Systems. WDL Publications, Canada.

Baranauskas, M. Cecilia C., Schimiguel, Juliano, Simoni, Carlos A., Medeiros, Cláudia B., 2005. Guiding the Process of Requirements Elicitation with a Semiotic-based Approach – A Case Study, HCI – International USA.

Booch, G., Rumbaugh, J., Jacobson, I. Unified Modelling Language User Guide. Ed. Addison-Wesley, 1999.

Bull, G., 1994. Ecosystem Modelling with GIS. Environmental Management. 18(3):345-349.–438.

Cachero, C. Una extensión a los métodos OO para el modelado y generación automática de interfaces hipermediales. PhD Thesis. Universidad de Alicante. Alicante, España. Enero 2003.

Durán A., Bernárdez, B., Ruiz, A., Toro M. A Requirements Elicitation Approach Based in Templates and Patterns. Workshop de Engenharia de Requisitos. pp.17-29 . Buenos Aires, Argentina. 1999

Durán A. Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información. Ph. Tesis. Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla. Sevilla, 1999.

EMF Technologies. Available in www.eclipse.org/modeling/emft/. Accessed in February 2011.

Escalona, M. J., Aragón, G. 2008. NDT. A Model-Driven approach for Web requirements. IEEE Transaction on Software Engineering, 34(3), 370-390.

Escalona, M. J., 2004. Modelos y técnicas para la especificación y el análisis de la navegación en sistemas software. PHD Thesis. Escuela Técnica Superior de Ingeniería Informática. Universidad de Sevilla.

Garzotto, F., Schwabe, D. and Paolini P., 1993. HDM-A Model Based Approach to Hypermedia Application Design. ACM Transactions on Information System, 11 (1), pp 1-26.

Kim, Do-Hyun, Kim, Min-Soo, 2002. Web GIS service component based on open environment. Geoscience and Remote Sensing Symposium, IGARSS '02. IEEE International Volume 6, Page(s):3346 - 3348 vol.6.

Liu, K., 2000. Semiotics in Information Systems Engineering. Cambridge University Press, Cambridge.

Macário, Carla, Medeiros, Claudia, Senra, Rodrigo, 2007. IX Brazilian Symposium on GeoInformatics - Geoinfo 2007 p.239-250.

M.J. Escalona, A.H. Torres, J.J. Gutiérrez, E. Martins, R.S. Torres, M. Cecilia, C. Baranauskas. A Development Process for Web Geographic Information System A Case of Study. ICEIS 2008International conference on enterprise information system; España (2008). Vol. HCI, pp. 112-117, ISBN/ISSN: 978-989-8111-40-1

Rossi, G., 1996. An Object Oriented Method for Designing Hipermedia Applications. PHD Thesis, Departamento de Informática, PUC-Rio, Brazil.

UML, Unified Modeling Language. http://www.uml.org/.