# ABSTRACTION OF COMPLEX SPATIAL DATA IN OBJECT-ORIENTED ENVIRONMENT

**COSTACHE GABRIEL**

*University Politehnica of Bucharest*

**Abstract** Geographical Information Systems (GISs) represent a powerful tool for capturing, storing, manipulating, and analyzing geographic data. This tool is being used by various geo-related professionals, such as surveyors, cartographers, photogrammetrists, civil engineers, physical planners (urban and rural), rural and urban developers, geologists, etc. They use the tool for analyzing, interpreting, and representing the real world and understanding the behavior of the spatial phenomena under their respective jurisdictions. Almost all of the systems used by the geoinformation community to date are based on two-dimensional (2D) or two-and a half-dimensional (2.5D) spatial data. In other words, one may find difficulty processing and manipulating spatial data of greater dimension than 2 in the existing systems, resulting in inaccurate or at least very incomplete information. In geomatics or geoinformatics we consider real world objects exist in three-dimensional (3D), thus it is desirable to have a system which is able to store, handle, manipulate, and analyze objects in a 3D environment.

**Keywords**: Geographical Information Systems, 3D spatial data, Geo-spatial Model, object-orientation environment

## 1. INTRODUCTION

Next generation of Geo Information System (GIS) requires a new way of spatial data modelling. We call the next generation of GIS 3D GIS. Fundamentally, a new digital model has to be established. Exploiting digital computing technology to improve the quality of life, or to prevent or mitigate hazards or disasters, would first require the construction of a model in digital form of the part of the earth and its environment.

A model distinguishes objects an object, or a set of objects, comprises the elements of reality under investigation. Spatial aspects are those related to shape, size and location that pertain to geometric properties. Non spatial aspects include name, color, function, price, ownership, and so forth, often referred to as thematic properties. Spatial aspects of reality can be well and economically represented in the form of graphics, whereas non spatial aspects, in many cases, can better be represented in text. Graphic representation facilitates rapid understanding of the situation in reality, permitting high level abstraction or description about neighboring relationships, while the textual representation is more suitable for aspects that cannot be graphically described. A digital model must be capable of relating these two representations. True representations and spatial information, for example sub-surface 3D objects, could not be successfully achieved with 2D systems.

## 2. LOGICAL DESIGN OF GEO-SPATIAL MODEL

The logical design defines all the data elements needed for the representation of each spatial object. It outlines the method for the translation of the data model from the conceptual level to a logical level. Three kinds of

logical data models are commonly used in information science: hierarchical, network, and relational. The hierarchical structures are those tree-like structures where a record is linked to another record via a 'parent and child' relationship. Based on the strict ordering of relationships, the structure is an acyclic directed graph in which no recursive relationship is allowed. Less strict than the hierarchical is the network structure, because it allows any kind of link between data elements, provided there are elements at both ends of the link. The operations using network data structure are fast and efficient, but the design, implementation and maintenance of this data structure are rather difficult.

The relational structure is more closely related to natural ways of thinking. Regardless of performance, relational structure offers the quickest and the easiest way to logical design. Another important logical design which has become popular during the last decade is based on the object-oriented approach. The object-oriented approach extends the hierarchical and network structures by encapsulating related operations as additional attributes within each record. Logical design based on the object-oriented approach is rather involved, since we must also take into account the related operations.

Although a relational database approach yields several advantages, certain important aspects are still lacking, which the *object-oriented approach promises to fulfill*:
1/ The joint operation on several large tables causes a long response time. The object-oriented approach includes the hierarchical and network data structures that can efficiently represent topology and facilitate navigation among elements in the database.
2/ A more complete and precise control over each individual object
3/ Re-usability and extendibility of database management API (through the inject- oriented compiler) with no modification to the protected source code

If for example a user is decided to represent a road as a line-feature, it would not be possible to represent the road later as a band, as might be needed for abstraction on a larger scale, since the band is an area-feature. The whole hierarchy would have to be redesigned for every different level of abstraction, which could result in many classes. This approach may therefore be regarded as an *ad hoc* solution.

This approach does not fix the geometric representation of a feature at the design stage. It divides the components of a feature into two hierarchies. The inheritance hierarchy is used for the thematic attributes and the aggregation hierarchy for the geometric attributes. These two hierarchies are only combined at runtime (the construction phase of the spatial model), thus allowing the user to select different types of geometric representation for a feature.

## 3. OBJECT-ORIENTED APPROACH

The object oriented approach provides mechanisms allowing economic reuse of a computer code by extending the code to accommodate additional types of data without a major reprogramming effort. Moreover, it also provides abstraction mechanisms to natural model spatial objects.

**Inheritance:** For example, a city may be represented as a point, or as an area feature highly dependent on the user. To make the logical design flexible, the type of representation can only be decided upon during the construction of the spatial model.

**Aggregation:** For example, a line feature is a composite object consisting of a list of nodes. These nodes must be arranged in proper order; each pair of nodes defines a straight line segment that constitutes the geometry of the line feature. If the nodes are badly arranged, or if one of them is missing, the line feature cannot be correctly constructed.

**Association:** We can consider the case of a line feature as an example to decide whether association or aggregation of a set of arcs is more suitable. In the spaghetti model, where the relationship between any two consecutive arcs of a line feature is not considered important, the line feature may be treated as a collection of arcs. Here, the association may be adequate. In a topological model like SVVM, where traversing along the line

feature to support network analysis should be possible, aggregation is more suitable because the relationships between two consecutive arcs are also considered important.

**Polymorphism:** For example, an area feature and a line feature can have the same attribute 'length.' Length, as an attribute of the area feature, is the perimeter of the area boundary, whereas it is simply the length of the line feature.

### 3.1 Object-oriented Definition of a Spatial Object

The abstraction of real world objects consists of two parts: geometric and thematic. The geometric component contains information about shape, location and topology, while the thematic component contains human knowledge about other properties of the object (color, name, ownership, function, and so forth). In an object oriented approach, the geometric and thematic components are realized as objects that can be tied together by a 'feature object' through the encapsulation mechanism. The term 'class' is an abstract data type (ADT), whereas the term 'object' is used to refer to an instance of a class in object oriented programming terminology. The terms class and ADT are used interchangeably.
The representation of a real world object can be translated into three ADTs :

#### *3.1.1 Thematic class*

The ADTTheme class defines a data structure for the storage of thematic information which is highly related to the application domain in geoinformation, such as land use, transportation networks and water bodies. This class consists of information about common attributes and behaviors of descendant thematic objects. The purpose of having this class is to facilitate the definition of an inheritance hierarchy, minimizing redundancies and allowing re-usability between thematic information.

**Geometric class:** The geometric description of a spatial object is stored and maintained in the ADTGeometry class. This class defines a hierarchy of geometric primitives which comprise the geometric descriptions of a spatial object. The class provides a general data structure for the storage of components, describing the shape of each feature, its georeferencing scheme and its topological relationships with other features. One important aspect is that every geometric object has to be referred to its ADTFeature class. An ADTFeature object is akin to the boss who represents and rules the group of subordinates. This assumption helps us make the organization more natural and efficient.

**Feature class:** The ADTFeature class provides the interface between the users and the system. The class is also the entry point for the user to retrieve or store all components of the feature. In other words, both ADTGeometry and ADTTheme objects form 'part-of' an ADTFeature object.

### 3.2 Specialization of Classes

As we have said, each of the above classes can be further refined as more detailed objects. The following sections show the construction of class hierarchies.

**Thematic Hierarchy:** The class ADTTheme can be specialized as various subclasses, such as road, railway, river, control point. The construction of this hierarchy is very subjective, depending on the user's point of view and application.

There are, however, many advantages in modelling thematic information using an object-oriented approach, especially when an object has to be related to several themes at the same time. The object-oriented approach provides a straightforward solution to this representation through the multiple inheritance mechanism. A class can inherit properties from more than one parent class. Such a class then represents a combination of themes.
An example, taken from Figure 1, is the TRiver class which can be seen as 'is-a' TWaterBody and TNatural transportation network at the same time. By aggregating an object that belongs to the thematic hierarchy (for example, class TRiver) as the component of an object that belongs to the feature hierarchy (for example, line feature class), the object belonging to the feature hierarchy automatically carries multiple thematic information.

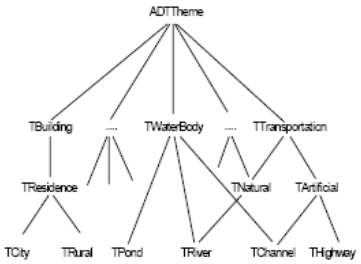


Fig. 1 Thematic class hierarchy (e.g. Egenhofer [1])

**Geometric hierarchy:** The class ADTGeometry can be specialized for each geometric primitive or simplex—node, arc, triangle, tetrahedron and so forth, as shown in Figure 2. These classes inherit properties from the parent class DTGeometry; each of them contains only the additional status and behaviors that are different from its ancestor.

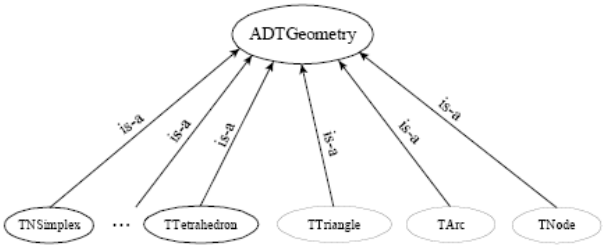


Fig. 2 Geometric class hierarchy.

Tnode : 0D GCRSO, Tarc : 1D GCRSO , Ttriangle : 2D GCRSO , Ttetrahedron : 3D GCRSO (GCRSO: "geometric component of the representation of a spatial object"

**Feature Hierarchy:** The class ADTFeature is specialized into four specific classes: point, line, area and body, as shown in Figure 3. Each derived class has its specific behaviors and attributes in addition to the behaviors and attributes of the parent class ADTFeature. A simple example is the draw operation. Drawing a point may only require drawing a pixel on a screen, while drawing a line, or an area, requires additional operations. The topology has to be used to navigate in the database to obtain all the nodes and their links before the pixels can be drawn along the line, or along the boundary of the area. The specialization also helps streamline the handling of the geometry and topology of each particular subclass. The design of related functions can be concentrated on specifically for each one in turn, with no fear of their interfering with each other, even if the functions of the different objects have the same function names.

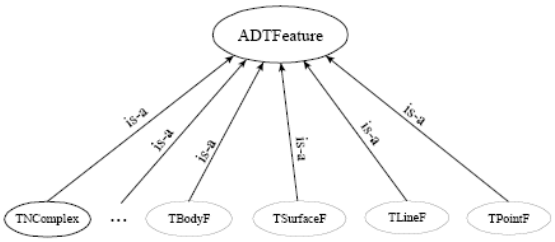


Fig. 3 Feature class hierarchy

Descriptions of the features: TpointF = 0D RSO, TlineF = 1D RSO, TsurfaceF = 2D RSO, TbodyF : 3D RSO, where RSO is **"**representation of a spatial object"

### 3.3 Aggregation of Objects

The ADTFeature class forms an aggregation hierarchy by taking objects belonging to the geometric and thematic hierarchies as its components (see Figure 4). This is a stage of assembling or manufacturing an instance of the

ADTFeature class. Subclasses of this class, for example, TPointF, TLineF, TSurfaceF, TBodyF, are also of aggregate types; a TLineF object may consist of many TArc objects.
For each subclass of the ADTFeature, the actual aggregation has to be done at runtime. This is because it is not possible to know at the design phase which specific class in the thematic hierarchy will be its thematic component. The dynamic referencing mechanism is the solution to this problem. The technique is first to define the aggregation, using the reference to a generic class (ADTTheme). During runtime, the user selects the more specific class (for example, class TRoad). Dynamic inheritance and aggregation take place here. The class that aggregates the class TRoad into the class TLineF is, in fact, derived at runtime. The TLineF object knows at that moment that its thematic component
is of the specific class TRoad, which is the descendant instead of the generic class ADTTheme. The reference to class ADTTheme is then changed to class TRoad.

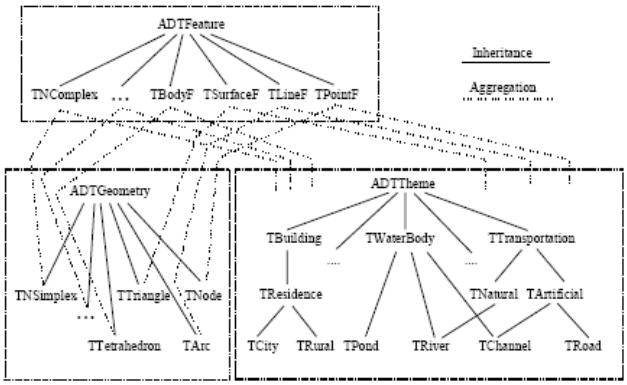


Fig. 4 Relationship between class hierarchies

### 3.4 Creation of Objects

In practice, users should first define their own thematic hierarchies according to the purpose of the application. For example, if the geoinformation is to serve the management of a road network, the thematic hierarchy should start from ‘general road’ and then specialize down to ‘primary road’, ‘secondary road’, ‘highway’, ‘superhighway’, and so on. The ADTFeature objects are created next. Every instance of this class must be registered into the container of ADTFeature. The user defines which theme is to be represented by which kind of feature. The notions of scale and resolution govern the choice. For example, an application using small scale maps may represent towns as point features (represented by TpointF class), while on a larger scale they may be represented by area features (represented by TSurfaceF class). To comply with this presumption, the definition of ADTFeature class must consider the specialized classes of ADTGeometry and ADTTheme.

The ADTFeature object created at this stage has to be considered incomplete, because of the lack of geometric content (see Figure 5). Completion can only occur when the reference to the geometric container has been established and the geometric container filled with all necessary references to ADTGeometry objects.

The ADTGeometry objects are the last to be created. The reason for this is that ADTFeature objects are not georeferenced before the stage of data acquisition. The specialized class of ADTFeature defines the specialized class of ADTGeometry object to be captured. When the user decides that a river will be represented by a line feature, the ADTGeometry objects to be captured are of the TArc class, and certainly not of the TTriangle class. Because the TArc object has references to two TNode objects, the user is forced to capture (create) two TNode objects prior to the creation of the TArc object. This engenders strict discipline in collecting data with expected high consistency.

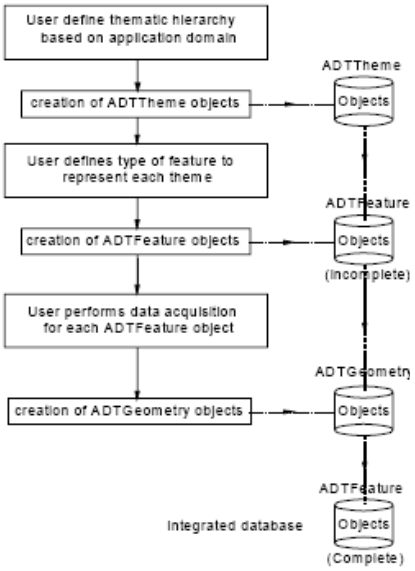



Fig. 5 Steps to creating objects.

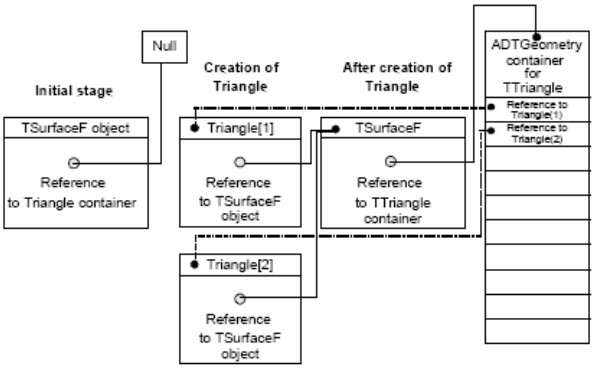



Fig. 6 Referencing scheme

**4 OBJECT-ORIENTATIONS OF TINs SPATIAL DATA**

This section explains the development of an OO database for TIN (triangular irregular network, e.g Abdul-Rahman [5]) data using a commercial database management system, POET OO DBMS (e.g. POET Software Corporation [4]).

The POET DBMS is utilized in this work as part of the OO spatial data modelling. It has: Encapsulation, Inheritance, and User-defined data type capabilities.

The DBMS is used to generate OO database from the constructed TIN spatial data. In this work, the schema needs to be modelled according to the POET database model; The PTXX compiler maps all the normal C++ classes into several relevant PTXX schema files which in turn are used for writing application programs (running under the normal C++ compiler) as well as for populating the database. The PTXX compiler also generates the OO database from the schema.

This built-in technique is adopted for this work: Object Query Language (OQL - detailed syntax POET(1996)). The following is an example of a query:
*defined extent allTEdge for TEdge;*
*select Edgefrom Edge in allTEdge*
*where Edge.EdgeAtr.EdgeName = "River*"*

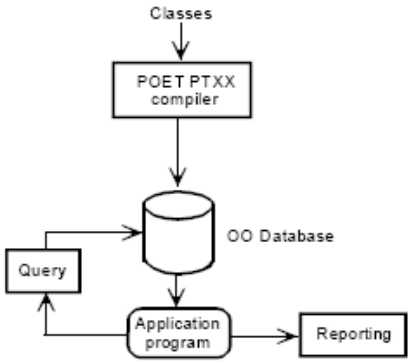



Fig. 7 The POET database development flow.

### 4.1 Object-oriented TIN-based Subsystems for GIS

The basic components of the system are data input processing, TIN data construction, TIN database, transformation operations, data output and user interface. Figure 8 shows the other major component of the proposed system - visualization, which uses the commercial software, i.e. ILWIS™ (Integrated Land and Water Information System) and AVS™ (Advanced Visualization System). These two packages are only used for validating the output from the rasterization process. The DBMS package is for the development of the OO TIN spatial database.

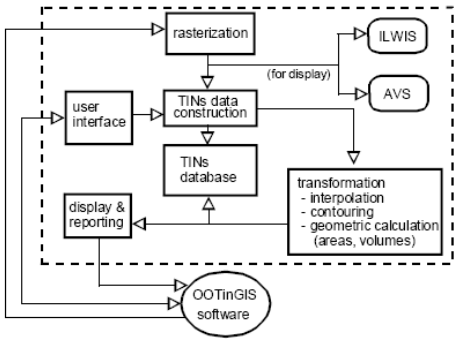



Fig. 8 The system for the TIN-based spatial data.

2D and 3D TIN tessellations are shown to work perfectly in the OO environment. The approach described in this chapter can be implemented in the development of TIN-based GIS system. The graphic output of the tessellations shown in Figure 9 clearly demonstrates the workability of the OO technique.

## 5. THE WEB AND 3D GIS

Applications based on network environments have already shown great potential in relation to geo-information. Examples can be online city maps and finding places (respectively routing) between points

Professionals involved in urban and landscape planning, cadastre, real estate, utility management, geology, tourism, army, etc. are especially keen on taking advantages of the third dimension. Since real world spatial objects are in 3D, it is obvious to extend GIS to the third dimension as well. However, the acceptance of 3D applications depends heavily on its profits. Therefore, we can say that the number of users could increase with the introduction of new and additional 3D functionality to the spatial system. Technologically, side, state-of-the-art computer hardware is already offering a reasonable means of dealing with the third dimension such as improved 3D visualization techniques. Among others, there is photo-realistic texturing, advanced lighting or real-time navigation that could attract more users to use such kind of applications. Within a Web-enabled environment, these principles of data input, management, analysis and representation are represented by or implemented within the components shown in Table 1.

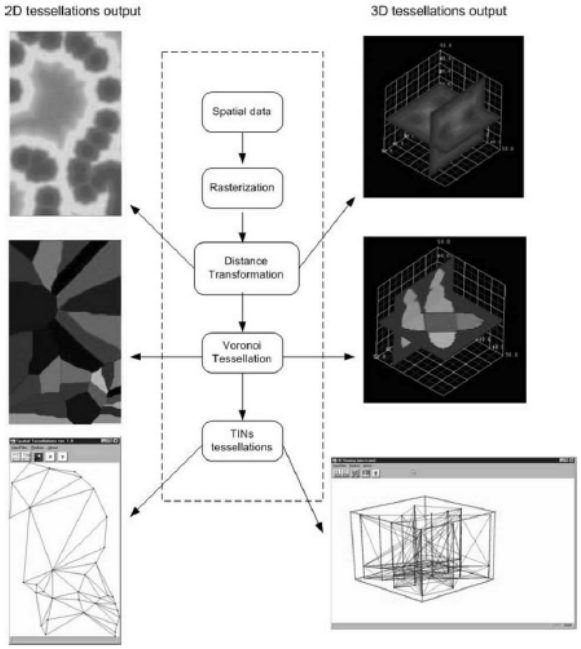



Fig. 9 The 2D and 3D TIN tessellations

GIS principles and their Web components. Table 1.

| **GIS Principle** | **Web Component** |
|---|---|
| Data Input | Client |
| Data Management | DBMS extended by a spatial component |
| Data Analysis | GIS Library on Server |
| Data Representation | Client/Server |

In order to achieve communication between the different components in a Web environment, a Web server is needed. The system should cover a complete GIS workflow within a Web environment. Figure 10 shows the general system architecture which is mostly "Client-Server".

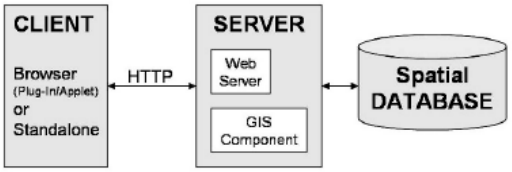



Fig. 10 A typical Web GIS minimum architecture.

The Client is an application, which can communicate with the Server through a standard Web protocol, for example HTTP. This application can either be in the form of a Web browser or stand-alone utility. In order to view and interact with GIS data, the browser needs to be extended by using an adequate Plug-In, Java Applet or both. Instead, a stand-alone application can be used, for example any GIS which supports the appropriate protocol to access other computers in the computer networks. The Web server is responsible for processing the request from the client and delivering the corresponding response. In Web GIS architecture, the Web server also communicates with the server-side GIS component. Moreover, server-side components are responsible for the connection to the spatial database, such as translating queries into SQL and creating appropriate representations

to be forwarded to the server. In reality, GIS components are like software libraries, which offer special "classes" to do spatial analysis on data. Besides the components, a very critical aspect is the functionality offered by the client or server side within Web-GIS. Figure 11 shows possible distributions of functionality for a client/server system based on the concept of the visualization pipeline (e.g. OGC [3]).

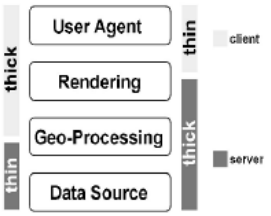



Fig. 11 Thin vs. thick within client server systems.

Figure 12 shows a standard request process—the page is requested via a browser. The request calls the designated PHP, which interacts with a database. The model given below explains the prototypical process.

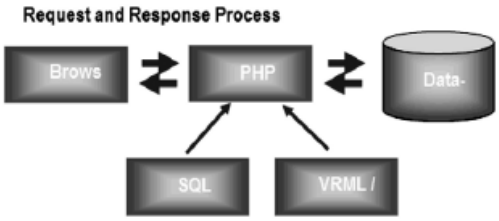
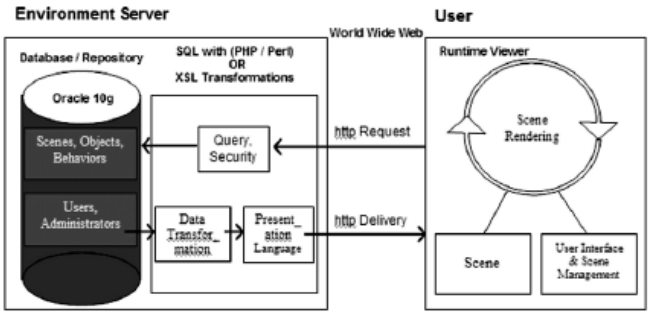



Figure 12 The request and response processes.

After receiving a response, the database sends the requested data to the PHP, which formats the data and sends the response to the requesting browser. In our case study, the data is returned to the PHP, which generates a VRML scene using the data from the database. On a client request, a connection is made to the DBMS and the spatial information of interest is selected from the DBMS and converted into X3D/VRML. A browser plug-in at the client side makes it possible to view the VRML or X3D output. VRML and X3D provide the possibility to start a script when a user clicks on an object. This functionality is used to retrieve the non-spatial information that is linked to a 3D geo-object. Via the VRML/X3D plug-in, a request is sent to an application server. The server receives and interprets the incoming information and sends a HTML with the required information back to the browser. For retrieving the spatial and non-spatial information from the DBMS, a technique is needed to communicate between a client and a database on a server. For this communication, several techniques are available such as ColdFusion, ASP.NET, ASP, JSP or PHP. The choice of the used technique is dependent on the Web server used.

The detailed architecture of publishing a 3D dataset on the Web is shown in the following Figure 13.




Fig, 13 Web publishing architecture of 3D datasets using 3D-GeoDBMS.

## 6. CONCLUSION

The current popular GIS software handles, manipulates, and analyses geographic data in 2D or 2.5D, thus using this system to manipulate 3D data full (particularly multiple Z coordinates) information about real world objects may not be appropriate. Therefore, the 2D GIS (or 2.5D GIS) need to be extended, i.e. to 3D GIS. Recent research in this domain have suggested that 3D spatial data modelling, structuring and data basing with object-orientation leads to better 3D GIS. This suggestion seems mainly arise from the complexity of 3D spatial data, as well as some positive features of object-orientation where every physical or spatial object of the real world can be defined during software development. It is therefore imperative to investigate the practicality of a means to improve the representation of natural objects in 3D and to manage them in an object-oriented GIS.

## 7. BIBLIOGRAPHY

[1] Spatial Data Moddeling for 3D GIS, Abdhul-Rahman, M. Pilouk, 2007, Springer
[2] Egenhofer, MJ, Frank, AU, Jackson, JP (1989) A topological data model for spatial databases. NCGIA Technical Report, No. 104
[3] OGC (2003b) OpenGIS Reference Model. (http://www.opengis.org/docs/ 03-040.pdf).
[4] POET Software Corporation,(1996) Why use an ODBMS. POET Technical References, http://www.poet.com/t_oovsre.htm#ODBMS
[5] Abdul-Rahman, A (1992) Triangular irregular network in digital terrain relief modelling. M. Sc. Thesis, ITC, Enschede, The Netherlands