# INTEGRATION OF GIS AND CAD AT DBMS LEVEL

**Shi Pu (1) and SISI ZLATANOVA (2)**

(1) ITC – International Institute for Geo-Information Science and Earth Observation
(2) Section GIS Technology, OTB, Delft University of Technology
The Netherlands

**ABSTRACT**

CAD and GIS have been designed for different applications but a tendency for convergence is increasing in the last years. For many different tasks integrations of GIS and CAD models are required. Currently there is no system that can easily integrate models from these two different domains. One of the bottlenecks in this integration is the supported data types. While CAD systems have a reach set of different types of shapes, GIS is limited to only points, lines and polygons. In this paper we suggest that DBMS can play an intermediate role in this integration by providing the missing data types. The two types of systems can selectively access only those data types that they understand. The consistency check is ensured at DBMS level.

## 1. INTRODUCTION

3D Geographic Information System (GIS) and Computer-Aided Design (CAD) models are getting closer than even before. Initially CAD systems were designed as 3D tools for modeling man-made 'things' (constructions, industrial parts, cars, etc.) in local coordinates systems. This intended as software for design and therefore a lot of emphasis was given to editing tools and effective 3D visualization. In contrast, GIS was designed to represent real world and more specifically all the tasks that used to be performed on paper maps. GIS, therefore, was able to maintain points, lines and polygons with geographic coordinates and corresponding attributes, and provide specific spatial analysis (very much application-oriented). Presently, the two systems offer increasingly similar functionality. CAD systems are evolving to Architecture, Construction and Engineering (AEC) systems with possibilities to work with 2D projections, define complex hierarchy of attributes and even perform GIS-like analysis. Similarly GIS users are demanding realistic 3D visualizations and exciting navigations over 3D models as in CAD (*Oosterom et al 2006*).

However, in this process of merging functionality, still many issues remain to be resolved. Three of the most appealing issues are:

- The types of supported primitives are significantly different. CAD software supports a broad range of primitives such as cone, sphere, cylinder and free-form curves (NURBS, B-Splines, Bezier), while these primitives are not present in the GIS world.
- The 3D editing tools of GIS packages are still very limited. Due to lack of real 3D primitives, 3D editing capability of GIS packages is still rather weak (*Zlatanova et al 2002*).

- The 3D topology, and thus 3D analysis, is lacking partially or completely in both systems (*Zlatanova and Stoter, 2006*).

A direct solution to this integration would be having a uniform data types both CAD and GIS models, and conversion between different formats can be naturally avoided (e.g. Breuning and Zlantanova, 2006). But currently this approach can be hardly implemented due to the huge differences between CAD and GIS data types and the file formats they use to store the models. Actually, even different CAD software use different formats, and so does GIS software.

An excellent option to resolve the problem of different file formats is the utilization DBMS for storage models designed in the two systems (*Zlatanova 2006*). The motivation behind is twofold. First, DBMS has already proved that this is a perfect system to maintain large data sets in secure and reliable way. Second, current mainstream DBMS (Oracle, Ingres, PostGIS, etc.) already support basic spatial data types: point, linestring and polygon. These data types are only 2D, but all the shapes can have the 3$^{rd}$ dimension, i.e. 3D primitives can be stored in DBMS. Very soon a real 3D primitive will be also available (*Arens et al 2005*). Additionally, many functions are also available on the supported data types. Almost all the GIS and CAD systems provide access to these data types. This means that, for example a model designed in CAD system can be stored in DBMS and later accessed and visualized in GIS (*Zlatanova et al 2003*). The only missing component in this concept is the lack of the variety of shapes as in CAD systems.

In this paper we present our work on extending the spatial data types support by DBMS with data types for freeform curves and surfaces. The data types are developed as user defined data types. Most DBMS offer several possibilities (using either high level languages as PL/SQL or programming languages as Java and C) to create user-defined data types including spatial data types.

The results of this research are presented in the following three sections. Section 2 elaborates further on the way DBMS can be used for integration of the two systems. Oracle Spatial 10g is used as example. Section 3 reports our development and tests performed upon Oracle Spatial 10g and two CAD systems, i.e. Microstation and AutoCAD. Section 4 discusses further research and shortcoming developments.

## 2. DBMS AS INTEGRATION MEDIUM

Since the 1990s, an increasing number of commercial DBMS provide spatial extensions. DBMS have already gained trust in robust management of large amounts of data. Many well-know characteristics of DBMS contributed to this process: multi-user control on shared data and crash recovery, automatic locks of objects during database transactions, advanced database mechanisms to avoid loss of data, data security, data integrity and operations that comfortably retrieve, insert, and update data. Spatial extensions also conform to these characteristics. Although current spatial DBMS (DBMS with spatial extensions) support only point, line and polygon, a 3D data type and other special data types (e.g. data type for laser-scan points, TIN) are under development. Spatial indexing and many spatial operations (at the moment still 2D) are competing with the functionality offered by traditional GIS systems.

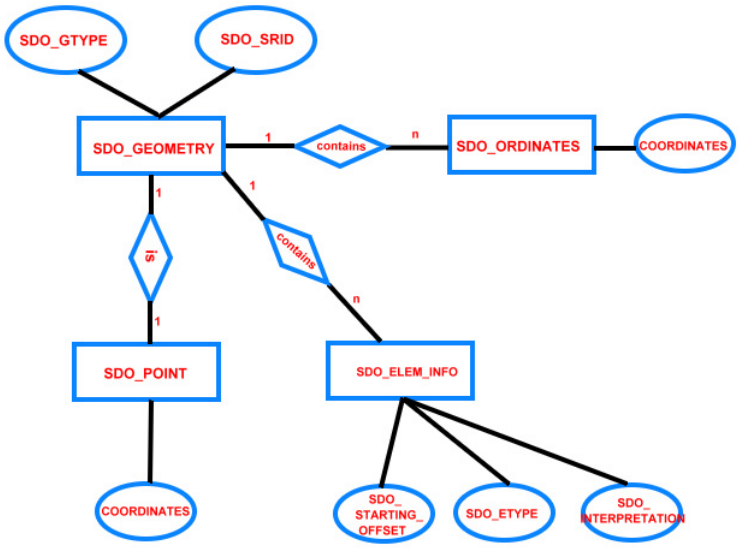


*Figure 1: SDO_GEOMETRY in Oracle Spatial 10g.*

Let us have Oracle Spatial 10g as example. Oracle Spatial 10g is one of the most powerful spatial DBMS on the market. It uses a uniform data type: SDO_GEOMETRY to represent all supported spatial data types (point, line/linestring, polygon, circle, etc.). Figure 1 gives the Entity Relation diagram for SDO_GEOMETRY, where different data types can be stored by setting different attributes. Some of the important attributes are (Oracle 2005):

**SDO_GTYPE** The SDO_GTYPE value is 4 digits in the format dltt. D is the number of dimension (2, 3, or 4); l specifies which dimension (3 or 4) contains the measure value; tt represents geometry types, for example, dl01 represents point, dl02 represents line/curve, dl03 represents polygon, etc.

**SDO_ELEM_INFO** This attribute indicates the format of coordinates in SDO_ORDINATES by repeating of the following attributes:

**SDO_STARTING_OFFSET** This attribute indicates the starting offset of current element's first coordinates in SDO_ORDINATES;

**SDO_ETYPE** Type of current element. For example, 1 for point related types, 2 for line related types, and 3 for polygon related types.

**SDO_INTERPRETATION** Interpretation for SDO_ETYPE. For example, with the same SDO_ETYPE value1, SDO_INTERPRETATION value 1 indicates a point, and SDO_INTERPRETATION value 3 indicates a point cluster with 3 points.

**SDO_ORDINATES** is the arrays of coordinates, which are stored there according to the format specified in SDO_ELEM_INFO.

Numerous GIS and CAD/AEC vendors provide extensions that use the data types offered by Oracle Spatial 10g. For examples, Bentley software (well-know with MicroStation) provides access in two ways, i.e. via the vertical application GeoGraphics and via a light java

application Spatial Viewer. Using both ways it is possible to retrieve spatial data stored as SDO_GEOMETRY in Oracle Spatial, and visualize them in MicroStation. Reverse operation, i.e. posting to Oracle Spatial is also possible. While Spatial Viewer can assess only spatial data types, GeoGraphics allows definition of features (i.e. spatial objects with attributes). The attributes in this case are also stored in the DBMS.

As it can be realized, GIS data types (points, line/linestrings and polygons) can be readily organized in DBMS. This is not true for CAD data types. CAD systems have more complex data types such as cylinders, cones, and freeform shapes and those cannot be stored in DBMS. With existing spatial data types in DBMS, CAD users have to convert complex data types to existing types before posting to the DBMS. Usually this conversion requires complex algorithms, calculation time and additional storage space. Furthermore it might happen that the data conversion is impossible for some data types. Typical examples of such complex shapes are freeform curves and surfaces.

If DBMS offers the missing data types as well as functions for conversion to simple data types (used by GIS), both GIS and CAD applications could benefit from the database storage. The following chapters present our first developments and initial results in this direction, i.e. freeform data types NURBS (Non-Uniform Rational B-Spline), B-spline and Bezier and their implementation in Oracle Spatial 10g.

## 3. CASE STUDY

In this case study, we implemented freeform data types (NURBS, B-spline and Bezier curves/surfaces) in Oracle Spatial 10g. Freeform shapes in two well-known CAD applications: MicroStation and AutoCAD, can be maintained in Oracle using our new data types. The tested models are a sport car and a building.

### 3.1 Freeform curves and surfaces

Although unfamiliar to GIS specialists, freeform shapes (curves and surfaces) have been extensively used in design of machines (e.g. cars), but also in geological modeling for representing underground surfaces and bodies. Modeling with freeform shapes is progressing in the AEC world as well. A variety of constructions (buildings, roofs, stadiums, bridges, etc.) are being modeled with NURBS, B-splines and Bezier curves and surfaces. All the large CAD/AEC vendors (Bentley Systems, Autodesk) also support them.

Let us consider NURBS as example (B-splines and Bezier could be derived from NURBS). The required parameters for defining a NURBS curve (Figure 2 left) include:

- Control points: used for the approximation of the curve.
- Weight value of each control point: indicating the “importance” of a control point.
- Degree: stating how “free” this curve can be; degree 1 indicates line string.
- A knot vector: give partition to the curve so that each control point only influences certain partitions of the whole curve.
- Trim value: optional, used to represent a part of the whole NURBS curve.

NURBS surface (Figure 2 right) can be described by similar parameters but in two directions. More detailed explanation about NURBS can be found in (*Piegl and Tiller W. 1997, Shi 2005*).

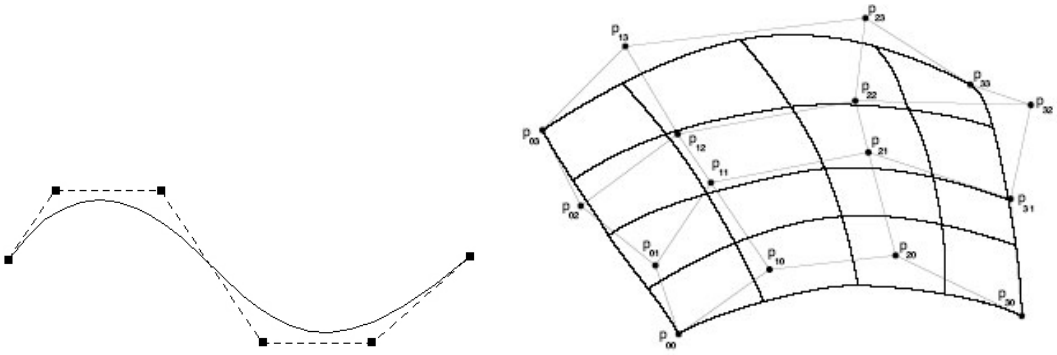


*Figure 2: NURBS curve (left) and surface (right).*

Some of the most interesting properties of NURBS are that they provide:

- a common mathematical form for both standard analytical shapes (e.g. cones, spheres) and freeform shapes,
- a flexible way of designing a large variety of shapes
- possibility for editing in the vicinity of one point (section)
- reasonably fast, numerically stable and accurate algorithms for evaluation of the shapes.

Although NURBS have certain disadvantages such as complex data structure, require more parameters in representing analytical shapes (cone, sphere) than common method, and etc., NURBS method is still widely used by CAD vendors.

As mentioned above, the other two shapes Bezier and B-splines are simpler compared to the NURBS. B-splines do not have weights for the control points. Bezier is the simples of the three having only control points and degree. They properties therefore also differ. Nevertheless, they are also widely used in 3D modeling and (due to their simplicity) supported by all CAD systems.

While developing our conceptual model, we have considered the fact that the tree data shapes can be derived from each other by eliminating some of the parameters. Therefore we have defined a parent class GM_SplineCurve having as attributes *controlPoints*, *degree* and *knots.* From this class the classes for the three freeform shapes can be derived. The concept for surfaces is in principle the same. Figure 3 shows the conceptual design of freeform curves and surfaces in DBMS, further details can be found in (*Shi 2005*).

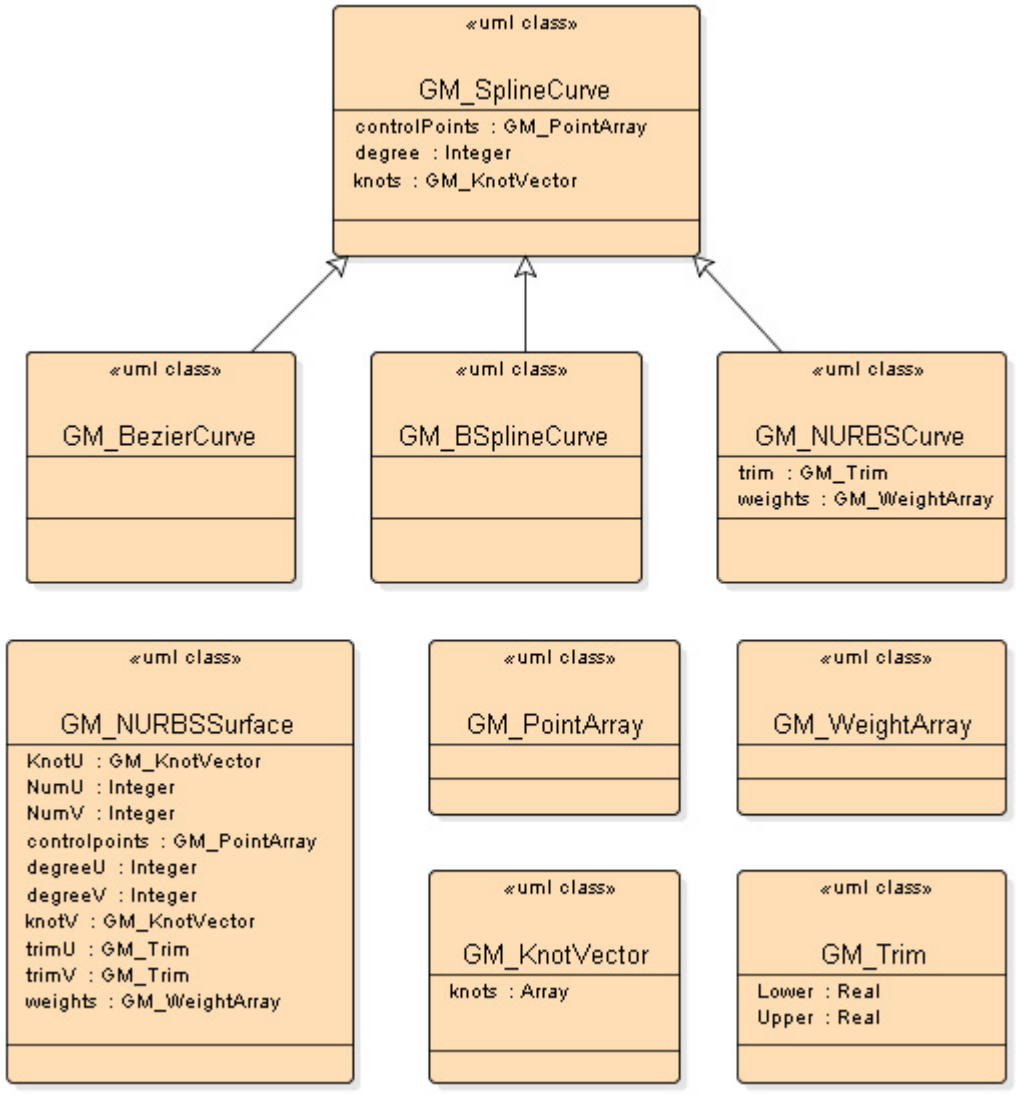


*Figure 3: UML model of freeform shapes.*

## 3.2 Creating user-defined data types

As mentioned above, Oracle enables users to create user-defined data types. With this convenient option, we have implemented NURBS, B-splines and Bezier in Oracle Spatial 10g as independent data types. This is in contrast to the philosophy of Oracle Spatial object-relational model where different data types are stored using one uniform data type (SDO_GEOMETRY).

User-defined data types in Oracle can be declared using the SQL statement CREATE TYPE (Oracle 2003). The implementation of the declaration can be PL/SQL, Java or C++. In our implementation, Java has been selected due to the good support of Oracle Spatial. PL/SQL is also a good candidate for new data types, but less efficient for mathematical and graphical computation than Java. The overall procedure for creating data types with using Java can be subdivided into 3 main steps:

- Java class creation;

- loading of the classes in Oracle spatial;
- and declaring the data type in Oracle using the SQL statement CREATE TYPE.

The mapping between Java classes and Oracle Spatial data types is quite straight forward: Java classes map to Oracle spatial data types as Java attributes map to data type attributes. The following SQL statement CREATE TYPE reveals the mapping:

```
SQL> CREATE or REPLACE type GM_NURBSCurve as object
external name 'GM_NURBSCurve' language java using SQLData(
degree number external name 'degree',
controlPoints GM_PointArray external name 'cpt',
knots GM_KnotVector external name 'knots'
);
SQL> /
Type created.
```

Further details on the implementation can be found in (*Shi 2005*).

### 3.3 Create functions on new data types

In addition to the data types functions (operations) are needed to manage the freeform data types in DBMS. The discussion which functions has to be available at DBMS level is not new (*Zlatanova and Stoter 2006*). We believe that specialized functions, which are of interest only for a particular application, have to be developed outside DBMS. Some very complex, time consuming functions might be also more suitable for including in CAD or GIS software. However, generic functions than can be used by many applications belong to the DBMS. A number of functions needed for validating the shapes and data consistency check are also important for a DBMS implementation.

In this research, we have also developed a number of user-defined functions on the new data types. The functions developed are relatively simple and aiming at demonstrating the use of the developed data types. They include functions for: validation, simple geometry transformations, conversion between different freeform shapes, computing distance between two freeform curves, and retrieving parameters of NURBS such as knots, control points and their number.

For example, the following SQL statement checks whether NURBS curve(s) from table TEST1 is valid:

```
SQL> select a.geom.validation() from test1 a;
A.GEOM.VALIDATION()
-------------------
1
```

Result 1 indicates this is a valid NURBS curve. This means that all the parameters are introduced and whether all the conditions that define the curve are correct. A set of very useful functions are the transformation functions. They allow a newly designed model (usually in a local coordinate system) to be transformed and placed in existing 3D GIS model. More examples of functions can be found in (*Shi, 2005*)

### 3.4 Exchange of models between different CAD packages

The last step in our developments was ensuring access to these data types from CAD software. We have performed tests with a car designed with NURBS and a building. Two CAD packages were used for this test MicroStation and AutoCAD. Additional software has to be

developed in both CAD systems, which reads the data types from DBMS and converts them to the internal representation of the two CAD systems.

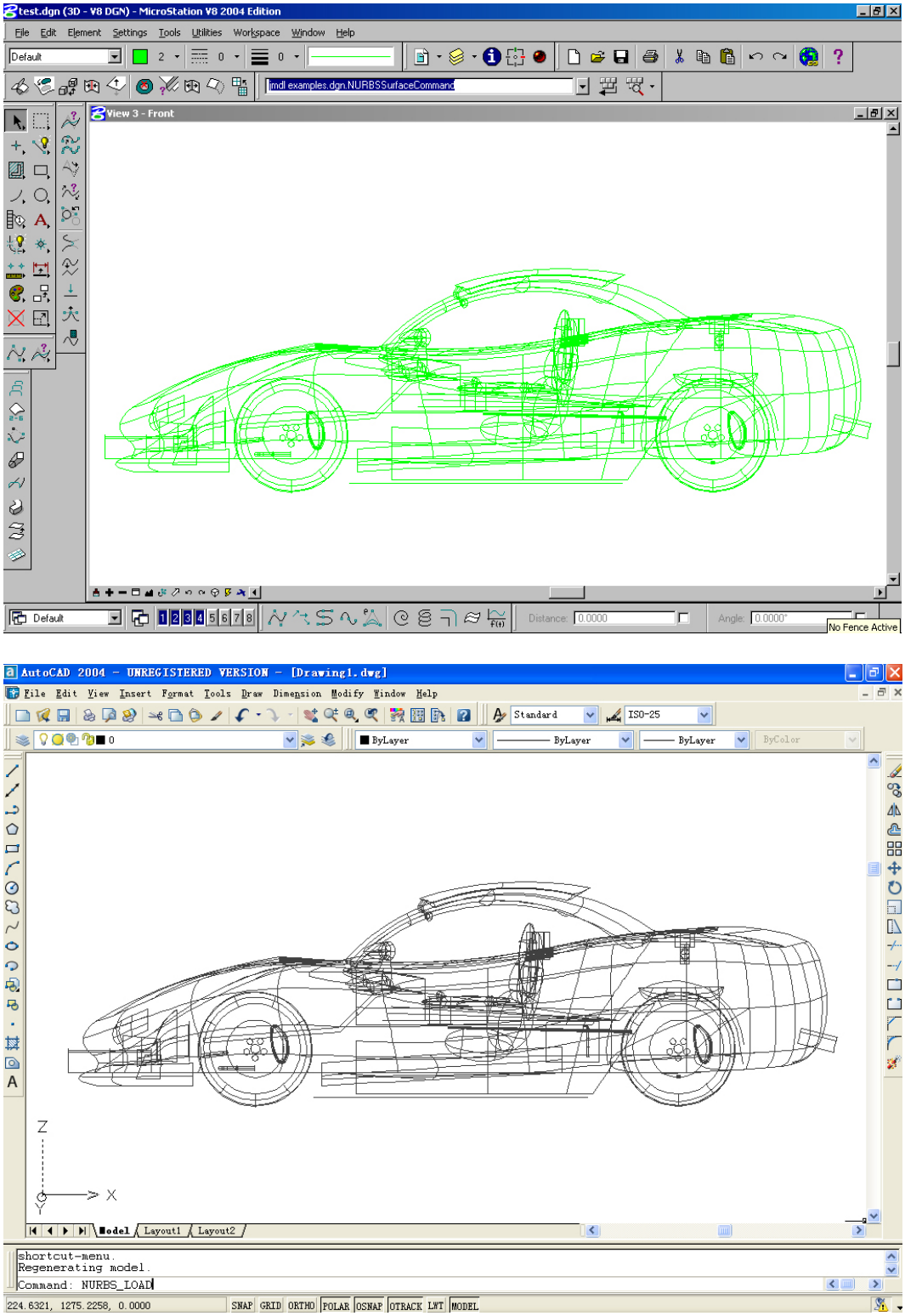


*Figure 4: Retrieval of NURBS model in MicroStation and AutoCAD.*

Figure 4 portrays the sport car modeled with only NURBS surfaces retrieved from the Oracle Spatial 10g and visualized in MicroStation. The same car was accessed and visualized also in AutoCAD without loss of surfaces. However, we have discovered some missing surfaces when we have exchanged the same car via DXF file format.

The second model was a building. It was a combination of linestrings, polygons and NURBS curves. The organization of this model was a bit more complex than the model of the car. While the NURBS were stored using the user-defined data types, the linestrings and the polygons were stored using SDO_GEOMETRY data types. Since the new data types are implemented outside the object-relational model of Oracle Spatial, they cannot be stored in one column with SDO_GEOMETRY data types. Therefore they were stored in two different tables. To be able to keep the relationships between the different parts of the building, all the components were given identifiers and were properly organized in a separate table.

The second model was also successfully exchanged between Oracle Spatial 10g and MicroStation without data loss.

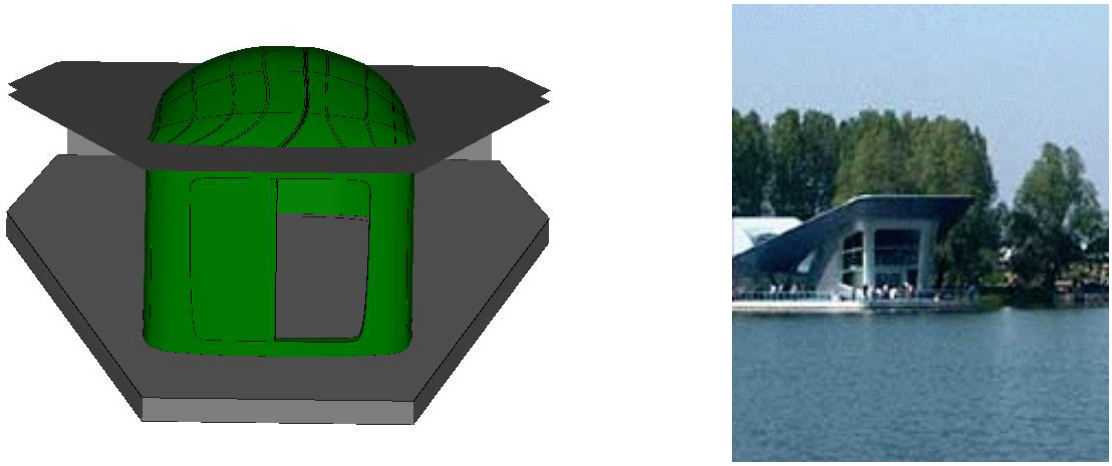

*Figure 5: The Netherlands pavilion at the Dutch Flower festival, Haarlemmermeer 2002: 3D model (left) and real building (right)*

## 4 CONCLUSIONS AND FURTHER RESEARCH

The experiments have clearly shown that geo-DBMS can be successfully used to manage complex data types and access (retrieve and modify) them form different applications. GIS models stored in DBMS can be easily integrated with newly designed buildings (or other constructions) and visualized and edited in CAD systems.

Despite the promising results, we have to admit that this research is still in its infant stage: a lot of issues need further investigations and developments.

The next very important step will be further investigation on efficient organization of 3D GIS data sets and CAD models represented with NURBS, B-splines or Bezier elements in one database. In the current implementation the table names were hardly coded in the software. However, the utilization of metadata indicating the nature of the data types would be quite beneficial for practical use.

Further research is also needed to derive strict rules regarding validity of NURBS. The implemented validation functions check only few of the freeform curve properties. It should be also decided which functions are worthy implementing at database level. Needless to say that many operations on freeform shapes are already available in CAD systems or might be too expensive for implementation at database level. For example, intersection of two NURBS requires complex computations, which may violate the performance of the database. It would be interesting to investigate which of the operations currently implemented for simple data types (such as union aggregation, areas, intersection, etc.) would be interesting and meaningful for freeform shapes.

The research reported here does not completely resolve the integration problem with respect to GIS systems. GIS software does not support NURBS and therefore the new data types would be still a problem for GIS software. However, the freeform data types can be modified to simple data types and at least visualized in GIS. Such conversions are mathematically possible and can be implemented at DBMS level in functions or in database views. For example NURBS surface can be approximated to a triangulated surface, which can be further represented by SDO_GEOMETRY. The smoothness of the shapes will be lost and it will not be possible to edit it (since these conversions are non-reversible) but the CAD model would be available for integration and visualization in GIS software.

Although the integration between CAD and GIS is not completely resolved, we consider our developments a significant contribution in this direction.

## REFERENCES

*Arens, C., J.E. Stoter, and P.J.M. van Oosterom*, *2005*, Modelling 3D spatial objects in a geo-DBMS using a 3D primitive, In: *Computers & Geosciences*, Volume 31, 2, pp. 165-177.

*Breuning, M. and S. Zlatanova, 2006,* 3D Geo-DBMS. in: *Large-scale 3D Data Integration: Challenges and Opportunities*, Zlatanova and Prosperi (Eds.), Boca Raton: Taylor&Francis, pp. 88-113.

*Oosterom P. van, J. Stoter, and E. Jansen, 2006*. Bridging the worlds of CAD and GIS, in: *Large-scale 3D Data Integration: Challenges and Opportunities*, Zlatanova and Prosperi (Eds.), Boca Raton: Taylor&Francis, pp. 9-36

*Oracle, 2003,* Oracle developer's guide 10g release 1.
Available on-line at http://www.oracle.com/technology/documentation, (accessed 02.01.2006).

*Oracle, 2005,* Oracle Spatial 10g User Reference.
Available on-line at http://www.oracle.com/technology/documentation, (accessed 02.01.2006).

*Piegl, L. and Tiller W., 1997*, The NURBS Book 2nd Edition, Springer-Verlag.

*Pu, S. 2005,* Managing Freeform Curves and Surfaces in a Spatial DBMS, MSc Thesis, TU Delft, 2005, 77 p. Available at http://www.gdmc.nl/publications (accessed 02.01.2006).

*Zlatanova, S. and J. Stoter, 2006*, The role of DBMS in the new generation GIS architecture. In Frontiers of Geographic Information Technology, S.Rana and J. Sharma (Eds.), pp. 155-180 (Berlin: Springer-Verlag, 2006).

*Zlatanova, S., T.P.M. Tijssen, P.J.M. van Oosterom and W. C. Quak, 2003*, Research on usability of Oracle Spatial within RWS Organisation, GISt No.21, ISSN: 1569-0245, ISBN: 90-77029-07-9, AGI-GAG-2003-21, Delft, The Netherlands, 75 p, http://www.gdmc.nl/publications (Accessed 01.02.2006).

*Zlatanova, S., A.A. Rahman and M.Pilouk, 2002,* Trends in 3D GIS development, *Journal of Geospatial Engineering*, 2, pp. 1-10.

## CVS OF THE AUTHORS

**Shi Pu** is a PhD student at the Earth Observation Science Department at the International Institute for Geo-Information Science and Earth Observation, the Netherlands. He has received his MSc at Delft University of Technology, the Netherlands in 2005. His main research interests are in geo-databases, computer-aided design and virtual reality.

**Sisi Zlatanova** is an assistant professor at Section GIS Technology, OTB Research Institute for Housing, Urban and Mobility Study, Delft University of Technology, the Netherlands. She has received her PhD at Graz Technical University, Austria in year 2000. Her main research interests are in 3D modeling, geo-databases, virtual reality and web GIS, and their applications for emergency response.

## CO-ORDINATES

Pu Shi
International Institute for Geo-Information Science and Earth Observation
Hengelosestraat 99, P.O. Box 6
7500 AA Enschede
The Netherlands
Telephone number : +31 53 487 4345
Fax number : +31 53 487 4335
E-mail address : spu@itc.nl

Dr. Sisi Zlatanova
Delft University of Technology
Jaffalaan 9
2628 BX Delft
The Netherlands
Telephone number : +31 15 278 2714
Fax number : +31 15 278 2745
E-mail address : s.zlatanova@tudelft.nl
Website : www.gdmc.nl/zlatanova