A NEW METHOD FOR INTERFACING 3D SIMULATION SYSTEMS AND OBJECT-ORIENTED GEO DATA SOURCES

M. Hoppen*, J. Rossmann, M. Schluse, R. Waspe

Institute for Man-Machine Interaction, RWTH Aachen University, Ahornstrasse 55, 52074 Aachen - (hoppen, rossmann, schluse, waspe)@mmi.rwth-aachen.de

Commission IV, WG IV/8

KEY WORDS: Simulation, Databases, GIS, Interface, Virtual Reality, City

ABSTRACT:

We present a new method to give 3D simulation systems access to object-oriented geo data sources. The goal is to dynamically adapt the simulation system to the data, allowing for the flexible design of GIS-based simulation applications by synchronizing the internal simulation database to the external data source. Synchronizing the internal to the external data schema makes both systems "speak the same language". Based on data synchronization, external object data can be transparently accessed by replicated instances in the simulation database afterwards and changes made to these instances can be resynchronized to the database. This enables the simulation system to flexibly adapt to different schemas and data, provides a caching mechanism for the data source and a persistence layer for the simulation system. In addition to this, the possibilities of a consistent multi-client operation as well as distributed simulation based on a central active database are currently investigated. From forestry to driving simulations several applications prove the methods' applicability and flexibility.

1. INTRODUCTION

Introducing state-of-the-art methods of geographic information systems (GIS) to a 3D real-time simulation opens up the field of GIS-based simulation applications. Such applications add dynamic aspects to previously static GIS worlds, which now can be "brought to life" using standard 3D simulation techniques - such as kinematics or rigid body physics - used e.g. in factory and driving simulations (Figure 1). A few projects (e.g. (Becker 1999), (Lindstrom 1997), (Wang 2005)) have already combined simulation systems with GIS but have not yet fully explored the capabilities of the resulting virtual reality-system and are usually only used for a specific field of application. We propose a more flexible approach for interfacing a 3D simulation system and object-oriented geo data sources. To sum up, the goal is to flexibly adapt the underlying simulation system structure to the data as well as to the data management system, not vice versa.

The design of this flexible interface is motivated by certain requirements:

- The geo data must not be converted offline to a simulation-friendly format before being used. This allows online access to a wide variety of GIS data sources and avoids data redundancy as well as loss of information.
- The interface must be able to flexibly respond to changes in the data source. Such changes include modifications to objects or even to the schema itself.
- Access methods must provide a high degree of **transparency** to hide the specific data source from the system.

Given this flexible interface, building up new simulations or adapting existing simulations to changes is simplified. Especially, methods known from the virtual commissioning of production lines (Rossmann 2007) can be adopted in the GIS context, e.g. to evaluate the impact of a new building on a cityscape. So called "GIS-based virtual testbeds" (Rossmann 2010) can be used for rapid prototyping approaches.

Figure 1. A GIS-based simulation application: A virtual drive through Stuttgart (data: City of Stuttgart)

Another design issue is the focus on object-oriented geo data sources that represent their content in terms of objects, attributes, relations and classes rather than tables. The object-oriented approach to data modeling has several advantages. Besides better fitting to modern programming paradigms, objects of the real world (e.g. a building) can be modeled more intuitively by defining appropriate classes with their relevant attributes. This approach enables intuitive world modeling rather than handling abstract data sets and provides the basis for our method of **semantic world modeling**.

* Corresponding author.

Providing the interface with read **and write** access to the geo data source, changes in the simulation system can be written back to the data source, which not only provides a persistence layer to the simulation system. In combination with an active data source it enables distributed simulations with multiple clients using the data source as a central communication hub.

The rest of this paper is organized as follows. In chapter 2 we introduce the 3D simulation system VEROSIM®, its internal database and our first example of an object-oriented geo data source. In chapters 3-5 we explain the synchronization process in general and schema as well as data synchronization in detail. Chapter 6 describes our current work in the context of active geo data sources and multi-client applications. We finish the paper with some examples of applications in chapter 7 and a conclusion in chapter 8.

2. SIMULATION SYSTEM AND DATA SOURCE

2.1 Simulation system

As mentioned above, the interface is defined between the internal simulation database and an external data source. In our case, we use the 3D simulation system VEROSIM®, which originally was developed as a robot simulation and virtual reality system. It is the basis for the development of new virtual reality based man-machine interaction methods and has applications for detailed planning of production lines in the field of automation technology and is the basis for a variety of driving simulators.

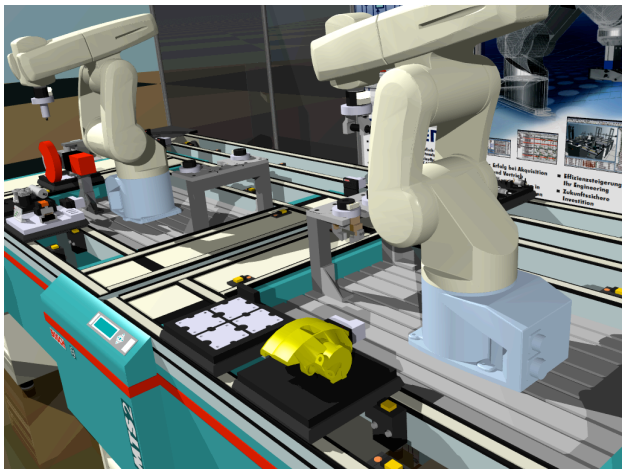

Figure 2. A work-cell with two robots and a conveyor belt

The flexible integration of GIS into the real-time simulation applications requires new methods to model the simulated environment. Therefore an important aspect of the system is its modeling concept. The system is based on an event driven and object-oriented graph database called VSD (VEROSIM® Active Simulation Database). It describes the components as well as their behavior and is designed to provide methods for parallel and distributed computing and visualization. An excerpt of the class hierarchy is given in Figure 3.

All our applications are based on object-oriented world models. In contrast to purely geometrical (the classical scene graph approach) or table representation of entities, the object-oriented model features a one-on-one modeling of real world objects as well as their hierarchy and relations in general. This approach inherited from robotic simulation models allows for the intuitive processing and simulation of objects, rather than abstract data sets.

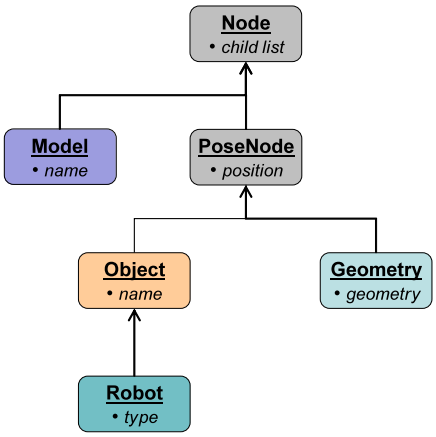

Figure 3. Excerpt of the simplified class hierarchy of the object-oriented graph database

As an example a small robotic work cell is shown in Figure 2 and the corresponding graph database for the work cell is shown in Figure 4.

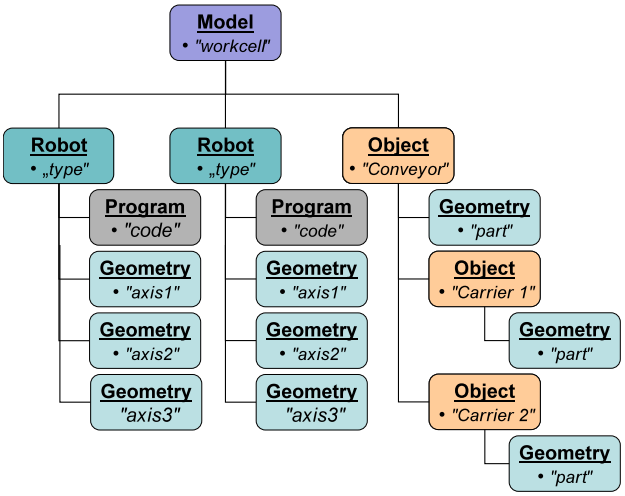

Figure 4. Excerpt of the simplified graph database of the work-cell illustrated in Figure 2

Furthermore the nodes provide mechanisms for interaction with the simulation environment – this is what distinguishes this database from standard scene graphs: The scene graph to render the scene is "just a view" onto the simulation database. Examples for such selective views onto the database are:

- Space partitioning can effectively be performed on the object nodes or any node derived from an object node, because the size of such a node depends on the attached geometries.
- The rendering engine only needs to consider the geometry nodes and not the rest of the model. Thus parts of the database appear as a scene graph to the rendering engine.
- The robot simulation part can be executed with only the robot nodes. All relevant information, such as input and output states, robot programs and kinematic chains are contained within these nodes.

With the aforementioned object-oriented data model, objects are provided "with a meaning" according to the real world – a concept we call **semantic world modeling**. Additional

geometrical, numerical and general data attributes of objects add further information to the objects and may be interpreted by software components. This is especially important to efficiently represent – and simulate – the behavior of these objects in the applications.

For example each tree/building in a forest/city is modeled as an instance of the corresponding tree/building class, while characteristics and connections are modeled as attributes and relations. The active parts of the simulated environments are modeled in the same way. A *Car* class describes the geometry as well as the behavior of a car driving through the GIS world while interacting (e.g. colliding) with other instances of the same class, the trees or buildings. An example for this approach is depicted in Figure 5. Here the properties of a tree object (geo position, type, height, diameter …) are visualized in a virtual environment using the metaphor of a "virtual business card".

When setting up a new simulation application, the developer only has to model object classes containing the properties and the behavior of the components new for that application. As an example integrating city models using the CityGML data standard (CityGML 2010) can be achieved by simply synchronizing the simulation system with a data source containing such data. This way, realizing a driving simulator (Figure 1) that combines standardized city models and rigid body simulation techniques is an easy task.

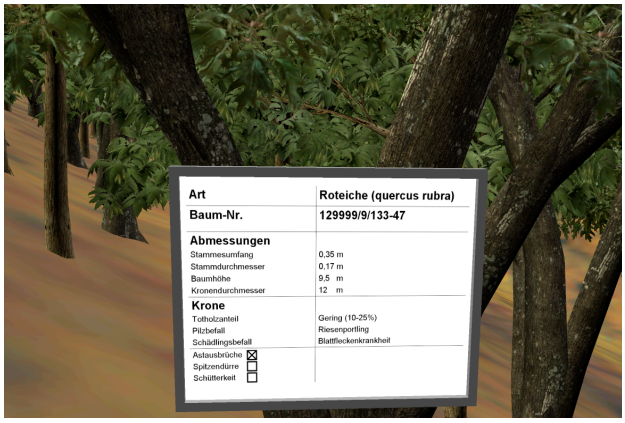

Figure 5. Example for semantic world modeling: the "virtual business card" of a tree

Another important aspect in the design of the database was the integration of the simulation functionality – the behavior of the components during the simulation process – into the database classes themselves. As an example, the robot node (Figure 4) contains all the logic to control a robot object.

To enable the synchronization process described in the next chapters, the VSD offers a **reflection API** and the ability to dynamically adopt new (sub-)schemas at run-time. In the VSD, objects and their attribute values are referred to as instances and properties. Each instance can have several properties of different types. For example, a *Building* instance can have properties *storeys*, *name* and *rooms*. The first two are value properties storing a single integer or string value. Other types are floating point numbers, enumerations etcetera. Multi-valued properties can also be defined. The third property is a reference property pointing at *Room* instances of which the building consists. The VSD allows single- or multi-valued reference properties that define 1:1 or 1:n relations between instances, thus defining the graph-structure mentioned above. Reference properties have well-defined target types, for example *room* can only point to instances of *Room* (or derived instances like *LivingRoom*). Reflection is provided by so called meta-instances describing classes of instances (name, inheritance, etc.) and meta-properties describing their attributes (name, type, multiplicity, etc.). This enables the simulation system to query meta-instances and meta-properties by name. **Dynamic schema generation** allows for the creation of new meta-instances and meta-properties at run-time.

2.2 Data source

As a first object-oriented external data source we are currently using the GML database management system SGJ (CPA, 2010). This object-oriented geo DBMS offers instantiation of a schema using XML Schema Definition (XSD). This allows for the definition of arbitrary classes, inheritance hierarchies, relations and attributes. SGJ can be controlled using its proprietary Java API with a kernel based architecture that allows full access to schema properties (classes, attributes and relations) and object data.

SGJ complies to several OGC standards. Data can be modeled conforming to GML, geometries are represented as Simple Features, Styled Layer Descriptors (SLD) can be used to build maps for its Web Map Service (WMS) and data can be read and written using its transactional Web Feature Service (WFS-T). Its standard compliance guarantees access to the data from different applications on different platforms. Thus, different tools on different platforms can be used to build up the GIS environment for a simulation and other clients than the simulation system can access the data. For performance reasons we use the Java API to access SGJ. Although until now our interface has only been tested using SGJ, the basic design is independent of actual chosen external data sources.

3. SYNCHRONIZATION MECHANISM

Given the requirements for flexibility identified in the introduction, we designed an interface with the key feature of a **dynamic two-level synchronization** (Figure 6, Figure 7). Not only object data is synchronized by replicating objects from the external data source to the simulation database and keeping them in sync. The data schema of the data source is also replicated and thereby instantiated in the internal database making both "speak the same language" and providing a common data schema for the whole system.

The system shows great flexibility towards changes in the data, i.e. to objects or even to the schema itself. This approach has several advantages over the direct usage of database objects. By using the (replicated) objects in the internal database, details of the actually used data source are hidden from all internal components of the simulation system (particles, physics, sensor data processing, rendering, etc.) allowing for transparent data access and real-time visualization and simulation. The internal simulation database works as a data cache to speed up repeating access patterns, which would otherwise lead to repetitive queries to the data source. Additionally, using an external data source provides the simulation system with persistence as changes made to replicated objects can be resynchronized back into the database. Despite these advantages, other clients or tools that are not realized using VEROSIM can still directly access the data source without invalidating the dynamic two-level synchronization as a whole.

4. SCHEMA SYNCHRONIZATION

The first level of this generic approach is the schema synchronization between the external data source and the internal simulation database. The result of this synchronization is a **schema mapping**: classes are mapped to meta-instances and attributes (or relations) to meta-properties (Figure 6). The schema synchronization can be realized in two ways: Either by schema matching or by schema transfer.

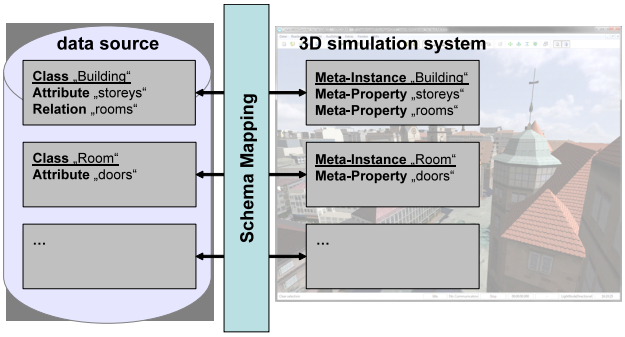

Figure 6. Exemplary result of schema synchronization: a schema mapping for a city data schema

4.1 Schema matching

In schema matching, a previously defined schema for the internal database is matched against the same schema in the external data source (e.g. class *Building* matches meta-instance *Building*, attribute *storeys* matches meta-property *storeys*, …). Technically the internal schema is hard-coded and defined at compile-time. The matching process itself is name- and type-based using VSD's reflection API mentioned above. Schema matching is mainly used for real-time critical and build-in functionality like geometry representation for rendering.

4.2 Schema transfer

Schema transfer is entirely performed at run-time using dynamic schema generation as described above. The data source's schema is evaluated and meta-instances are defined (e.g. a meta-instance *Building* is created) without being hard-coded. Based on the class's attributes and relations, the appropriate meta-properties are defined (e.g. meta-instance *Building* gets a meta-property *rooms*).

Of course, these two principles of schema synchronization can also be mixed, so that classes or attributes missing during the schema matching process can be added dynamically by schema transfer.

5. DATA SYNCHRONIZATION

After synchronizing the schemas, which is done only once during run-time initialization, data is synchronized on the object-level (including attributes and relations). This synchronization provides the basis for our basic principle of data access: transparency. No component of the simulation system has direct access to the data source – only the interface itself. Instead, these components access data by means of local copies in the internal database, i.e. replicated versions of data source objects. This built-in transparency provides for the flexibility, as data sources can easily be replaced by different ones, because they are completely encapsulated by the interface.

Based on this principle, data synchronization now includes object loading, management and unloading, change tracking and write back of changes (i.e. resynchronization). It spans the complete run-time of the system.

Object loading implies an object from the data source is replicated in the simulation database. This is realized by querying the object and instantiating an instance of the corresponding meta-instance (known from the schema mapping). Then all attributes and relations are copied to the corresponding properties and a mapping between the data source's object and the cloned instance is inserted into the **object mapping** (Figure 7). Hence after loading an object, the data it represents is transparently available as a local instance to the rest of the simulation system.

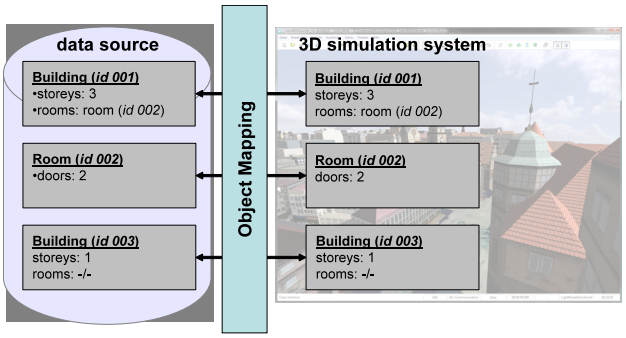

Figure 7. Exemplary result of data synchronization: an object mapping for city data

At any time, changes to the simulation database are tracked. This includes changes to instances as well as the creation and deletion of instances. Changes are queued as internal transactions and can be written back to the data source to resynchronize the internal and external state of data. Depending on the application, this resynchronization is carried out immediately or by user request. Resynchronization implements a **persistence layer** for the simulation system as changes to the virtual environment can be made permanent. This is a sustainable approach, as these changes are not applied to an offline-converted "simulation friendly" version of the data but to the original (geo) data itself. Thus simulation results can be saved and virtual environments can be manipulated permanently, right within the simulation system giving it more flexibility in the choice of possible applications (see chapter 7)

Besides loading, instances can also be unloaded, provided that no transactions have been stored for them. Unloading is simply realized by deleting the instance from the internal database and removing the object mapping. Usually, unloading is used for dynamically streaming parts of large data sets, e.g. city models. In the process of streaming, object sets are loaded based on certain filter criteria that include some kind of variable. This can be a filter with a spatial variable that is shifted from one section to the next to realize geometrical streaming, e.g. tile-wise loading and unloading of a city's buildings. Or the filter contains a restriction to the relation between objects realizing hierarchical streaming, e.g. (un-)loading all rooms of a building.

6. CURRENT WORK

One further aspect of data synchronization we are currently investigating is the integration of change notifications from the external data source, a precondition for multi client synchronization. Therefore an active data source is needed. We

describe this scenario as a two-tiered architecture with multiple simulation systems connected to a central active data source, e.g. a central geo database (Figure 8). This not only enables multi client operation on consistent data but also allows for the use of the data source as a communication hub and central data storage (Freund, 1999) that defines a common data schema by synchronizing it to all connected simulation systems.

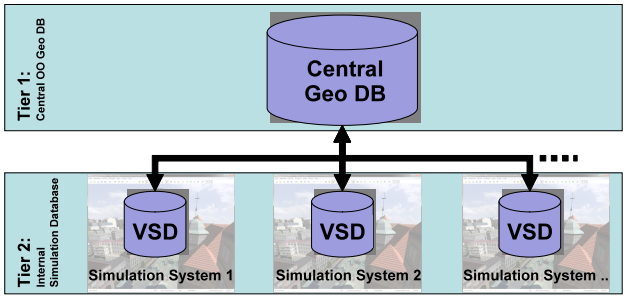

Figure 8. Two-tiered architecture with a central geo database and multiple simulation systems

In such a distributed simulation system, if changes of an object are written back (i.e. are resychronized) to the central geo database all other connected simulation systems are informed about these changes and apply them to their local copies of the same object accordingly. Figure 9 shows an example of such a communication sequence. A door is opened in the virtual environment on instance 1 of the simulation system, changing the object's property *open* to *true*. This change is resynchronized to the central geo database where the replicated door object's property is set to *true*. This change is notified by the central geo database to all connected clients, including instance 2 of the simulation system which sets the door's property accordingly, thus resynchronizing the distributed simulation as a whole.

To avoid notification loops, changes adopted from the central DB are not interpreted in the same way as changes from within the local simulation system. Thus, after simulation system 2 sets the door's *open* property to *true* it will not resynchronize this change to the central database.

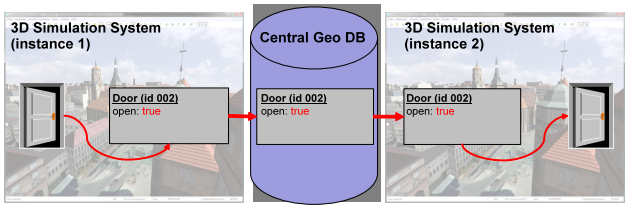

Figure 9. Example for a communication sequence via the central geo database: a door is opened

7. APPLICATIONS

Several GIS-based simulation applications using the interface introduced in this paper have been realized so far. One major application is the "Virtual Forest"™*, a science project of the German federal state of North Rhine Westphalia supported by the European Union (Rossmann, 2008). The goal of this project is to provide new means for forest inventory, serving as the basis for the development of new decision support systems and the application of state of the art industrial automation techniques to the forest industry. The project covers the acquisition, storage and processing of 240 million trees on the states total forest, covering an area of approx., 9,000 km². A visualization of the data of the "Virtual Forest" using virtual reality techniques allows for an efficient analysis of the impact of e.g. thinning operations (Figure 10) or storm damages.

Figure 10. A timber harvester navigating in the Virtual Forest

In the context of the "Virtual Forest", the proposed interface has not only been used for such simulations. Due to its flexibility we also based an integrated tool for forest inventory on our simulation system including the database interface proposed here (Figure 11). Proprietary data schemas based on the GML base schema were defined for inventory data, cadastral data or points of interest. The data is stored in different SGJ geo databases connected to the forest inventory tool and presented to the user as GIS layers. The user can load objects, change their attributes, create new or delete objects and write back the changes to the database using the resynchronization mechanism. The tool runs on desktop PCs or even ruggedized notebooks.

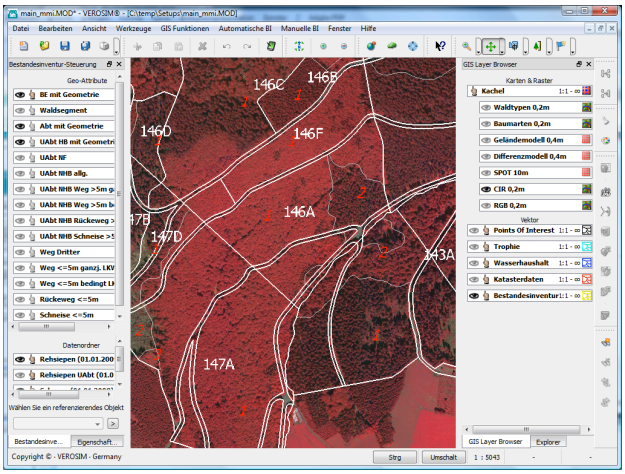

Figure 11. The Forest Inventory Tool of the Virtual Forest

Another field of application are 3D city simulations. Here, SGJ databases with CityGML (CityGML, 2010) models of cities like Stuttgart (Figure 1), Düsseldorf or Barkenberg (Kreis Recklinghausen) (Figure 12) have been connected to VEROSIM®. Another test dataset was available in the SEDRIS data format (SEDRIS, 2010) – our flexible approach let us

* The Virtual Forest is supported by the State of North Rhine Westphalia (NRW), Germany, the forest administration of North Rhine Westphalia and the European Union (Europäischer Fond für regionale Entwicklung - EFRE)

visualize the provided data and use it for simulations only by connecting the simulation system to the new dataset. Simulations include driving a virtual car (with physically correct behavior) through a city. Therefore the database containing the city model is connected to the simulation system using the proposed interface. After synchronizing the schema the virtual urban environment is partially loaded using data synchronization with geometrical streaming technology. Finally, a model of a car with dynamic behavior is placed into the scenario and the simulation is started. Other applications are virtual fly-throughs and data management in the virtual environment. For example, the resynchronization mechanism allows the user to change an object's properties from within the 3D simulation (Figure 12).

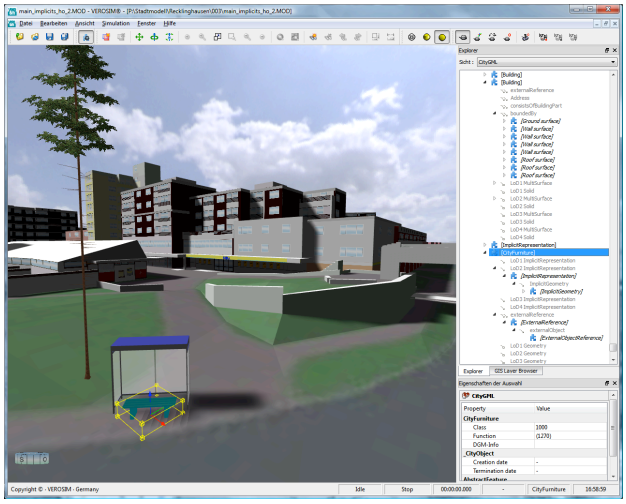

Figure 12. An object-oriented CityGML model and properties of a city furniture object (data: Kreis Recklinghausen)

8. CONCLUSION

In both applications – city and forest – the same database interface could be used for different data models (CityGML, SEDRIS, forest inventory, cadastral data, points of interest) without any adaptations. This shows the presented methods' adaptability to different fields of applications. In our opinion, they offer huge potential for many more, in single- and multi-client applications. For the future, we are already planning to use it for non-GIS scenarios like the distributed control and simulation of manufacturing processes.

REFERENCES

Becker, L., Bernard, L., Döllner, J., Hammelbeck, S., Hinrichs, K., Krüger, T., Schmidt, B. and Streit, U., 1999. Integration dynamischer Atmosphärenmodelle mit einem (3+1)-dimensionalen objektorientierten GIS-Kern. In: 13. Internationales Symposium "Informatik für den Umweltschutz", pp. 429-442, 1999, ed. C. Rautenstrauch, M. Schenk, Publisher Metropolis

Freund, E., Müller, M. and Rossmann, J., 1999. Data storage and flow control in automation systems by means of an active database. In: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA'99), Vienna, Austria, pp. 235-240, ISBN 90 5199 475 3.

Homepage of CPA Systems GmbH http://www.cpa-systems.de/ (accessed 6 August 2010)

Homepage of CityGML http://www.citygml.org/ (accessed 6 August 2010)

Homepage of SEDRIS Technologies http://www.sedris.org/ (accessed 6 August 2010)

Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L. F., Op den Bosch, A. and Faust, N., 1997. An Integrated Global GIS and Visual Simulation System. In: Technical Report GIT-GVU-97-07, Georgia Inst. of Technology.

Rossmann, J., Stern, O. and Wischnewski, R, 2007. Eine Systematik mit einem darauf abgestimmten Softwarewerkzeug zur durchgängigen Virtuellen Inbetriebnahme von Fertigungsanlagen. In: GMA-Kongress 2007 – Automation im gesamten Lebenszyklus, Baden-Baden, 12.-13. Juni 2007, VDI-Bericht Nr. 1980, pp. 707- 716, ISBN 978-3-18-091980-5.

Rossmann, J., Schluse, M. and Bücken, A., 2008. The virtual forest – Space and Robotics technology for the efficient and environmentally compatible growth-planning and mobilization of wood resources. In: *FORMEC 08 - 41. Internationales Symposium*, pp. 3-12, ISBN 978-3-9811335-2-3.

Rossmann, J., Schluse, M., Hoppen, M. and Waspe, R., 2010. GIS-Based Virtual Testbeds and their Application to Forestry and City Simulation. In: *Proceedings of the ASME 2010 World Conference on Innovative Virtual Reality (WINVR 2010)*, Ames, Iowa, USA.

Wang, X., 2005. Integrating GIS, simulation models, and visualization in traffic impact analysis. In: Computers, Environment and Urban Systems, Volume 29, Issue 4, July 2005, pp. 471-496, ISSN 0198-9715.