Dynamically Extending Spatial Databases to support CityGML Application Domain Extensions using Graph Transformations

ZHIHANG YAO[1] & THOMAS H. KOLBE[1]

Abstract: As a domain extendable 3D city model the OGC standard CityGML has been increasingly employed as a dominant consensus over different application areas. An essential challenge encountered in many research and commercial activities in the field of 3D city modelling is to find a reliable approach for achieving high-efficient storage and management of data models according to CityGML with its Application Domain Extensions (ADE) in order to ensure interoperable data access across broad application domains. Based on graph transformation, this paper introduces a new approach along with an extensive database structure that allows for dynamically extending the spatially-enhanced relational databases for handling arbitrary CityGML ADEs by means of graph transformation systems. With this approach, relational database schemas with simplified and optimized table structures can be automatically generated from the XML application schemas of CityGML ADEs by performing a series of user-defined graph transformation rules which can describe complex mapping rules for transforming object-oriented data models to relational database models in a fully declarative way. The proposed approach has been successfully implemented and tested based on a number of different CityGML ADEs like Energy ADE, Dynamizer ADE, and UtilityNetwork ADE.

1 Introduction and motivation

Over the recent years, the OGC standard CityGML has been increasingly used as a semantic 3D city model for describing the relevant entities in the landscape and urban space. It is designed as an application schema based on the Geography Markup Language 3 (GML3) and includes a wide range of feature classes along with their interrelationships as well as their relevant thematic and spatial properties which can cover the most varying needs of different application domains. However, in many practical applications, an essential issue often encountered while using CityGML is that many additional feature classes or extra attribute types are required to represent some certain objects or data structures for performing domain specific analysis or simulations such as energy demand calculations, utility network analysis, facility management, noise propagation simulations etc. For this reason, CityGML provides an extension mechanism called "Application Domain Extension (ADE)" allowing third parties to dynamically and systematically extend the existing CityGML data models by attaching extra application schemas which can contain additional feature classes and attribute properties defined for specific application domains. Since 3D city model objects usually have well-defined identifiers which are usually kept stable throughout the lifetime of the referenced real-world objects, the complete 3D city models being attached with diverse domain-specific information are hence considered as a good basis for

[1] Technische Universität München, Lehrstuhl für Geoinformatik, Arcisstraße 21, D-80333 München, E-Mail: [zhihang.yao, thomas.kolbe]@tum.de

building an integrative platform for realizing interoperable data exchange across different application domains.
In order to efficiently store and manage CityGML-based 3D city models regarding their complex data structures and spatial characteristics, spatially-enhanced relational database management systems (SRDBMS) were often chosen to serve as the central data repositories as they are usually featured with extensive spatial capabilities compared to other types of database systems. A representative relational database solution is the 3D City Database (3DCityDB), an Open Source software for the storage, management, and analysis of 3D city models according to the CityGML standards. It is shipped with a compact relational database schema resulted from a careful mapping of the object-oriented data model of CityGML 2.0 to a fixed database structure which has been employed as standard database implementation in many production environments and commercial projects all over the world. However, the 3DCityDB database schema has no support for CityGML ADEs and the underlying design decisions strongly rely on many manual steps, for example recognizing a certain complex model structure and mapping it to a particular target database structure, which make the automation of the mapping process for CityGML ADEs much harder than the generic solutions proposed in the many literatures. Furthermore, since CityGML ADE models can be arbitrarily defined with very complex model structures in practical situations, a challenging task is to find a way for dynamically and automatically extending the 3DCityDB database schema in order to efficiently store and maintain geospatial data of arbitrary CityGML ADEs (cf. Fig. 1).

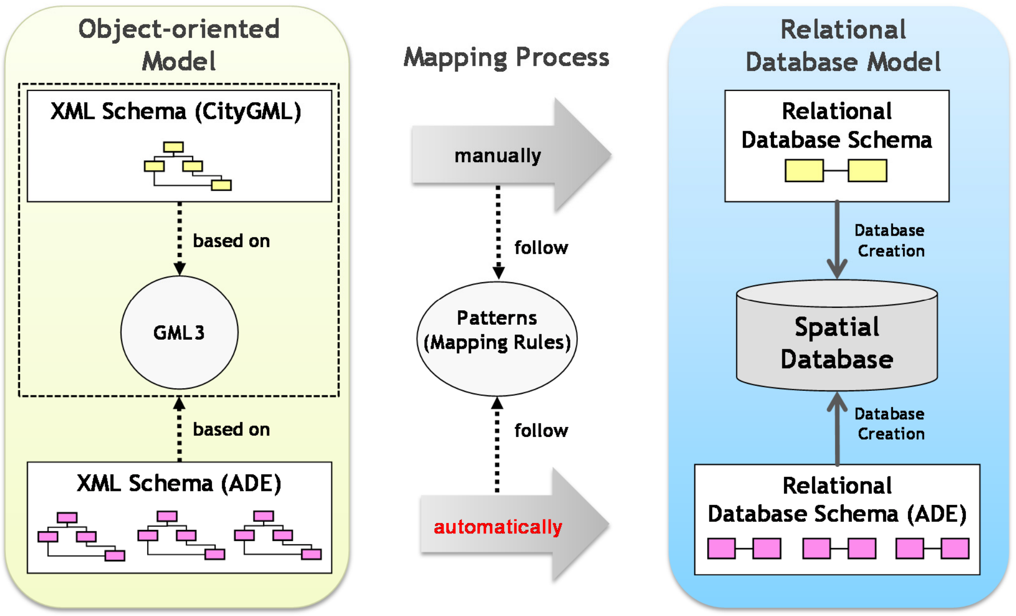

Fig. 1: Conceptual workflow for creating a dynamically extendable 3D geo-database for storage and management of CityGML ADEs using spatially enhanced relational database management systems

Within this paper, we will present a new graph-based framework which uses typed and attributed graphs for semantically representing the object-oriented data models of CityGML ADEs and utilizes graph transformation systems to automatically generate compact relational database

schemas extending the 3DCityDB. The transformation process is performed by applying a series of fine-grained graph transformation rules which declaratively describe user-definable mapping rules for transforming complex object-oriented data models to relational database schemas. With this approach, the 3DCityDB database schema has been structurally enhanced to be dynamically extendable for arbitrary CityGML ADEs and can, hence, be used as an integrative information backbone for interoperable data access for a wide range of domain-specific information according to the CityGML standard.
The rest of this paper is structured as follows: Section 2 addresses the relevant concepts for the modelling and development of CityGML ADEs and provides the theoretical foundation for our work. Section 3 introduces an extended database structure for 3DCityDB which becomes to be dynamically extendable for handling multiple CityGML ADEs simultaneously. In section 4 we propose a conceptual approach for schema transformation using graph transformation systems for realizing the automatic generation of relational database schemas from CityGML ADE application schemas. In addition, a technical implementation of the proposed conceptual approach is detailed in section 5. The last section draws the conclusions about the presented study and outlines the relevant aspects of our future research and development work.

2 Model Driven Approach for the development of CityGML ADEs

A fundamental understanding of the mechanisms behind the development of CityGML ADEs is the major prerequisite for the establishment of the graph-based framework presented in this paper. Basically, the data models in software and systems engineering are nowadays usually developed by following the well-known design approach called "Model Driven Architecture (MDA)" issued by the Object Management Group (OMG). MDA aims to provide a systematic way for facilitating the development of software and data models using Computer-Aided Software Engineering (CASE) tools (GA et al. 2006). The key idea of this design approach is that an abstract platform-independent model (PIM) representing the underlying conceptual models shall be first created at the earlier stage of the development lifecycle and then be implemented for different application platforms at a later stage by transforming it into the corresponding platform-specific models (PSM) such as XML schema and database schema etc. In order to ensure the unambiguity and consistency of the definition and description of model structures, the Unified Modelling Language (UML) was chosen as the standard modelling language for describing the platform-independent models since it comes with a rich set of graphic notations and syntaxes allowing to visually represent complex model structures as well as to fully represent the respective semantic meanings.
In the field of geographic information modelling, the combined use of MDA and UML has been predominantly adopted as the main instrument for the development of geospatial information models on the basis of the so-called "ISO 19100 standards family" issued by the Technical Committee (TC) 211 of the International Organization for Standardization (ISO). This bundle of standards comprises a series of abstract specifications which jointly provide a general guideline for geographic information modelling regarding the definition and description of geographic phenomena in order to ensure the interoperability of data exchanges across different application platforms. For example, the ISO standard 19109 "*Rules for Application Schema*" comprehensively specifies the rules on modelling of real-world features along with their attribute properties as well

as their interrelationships such as generalization/specialization and association relationships like aggregations and compositions (ISO 19109:2005). In particular, the spatial properties of features such as their geometric-topological characteristics can be comprehensively expressed using the geometry and topology models defined in the ISO standard 19107 "*Spatial Schema*" specification (ISO 19107:2003). Other relevant modelling aspects such as temporal schema, metadata, and coverage etc. are explicitly covered in the standards ISO 19108, ISO 19115, and ISO 19123 respectively. Based on these abstract standards, the ISO 19136 standard "Geography Markup Language (GML)" was released which provides an open and manufacturer independent framework for the definition of geospatial data models on the application level (ISO 19136:2007). The main GML model components like geometries, topologies, coverages, coordinate- and time reference systems etc. are wholly and partly drawn from the conceptual models defined in the ISO 19100 standard family and implemented as a XML-based schema (GML Schema) according to the XML encoding conventions specified in the ISO 19118 standard (ISO 19118:2011). Moreover, the ISO 19136 standard additionally specifies an extensive UML profile together with a normative UML-to-XML encoding rule set which builds an MDA-compliant framework allowing for the development of a variety of application schemas based on a standardized exchange interface using GML (cf. Fig. 2).

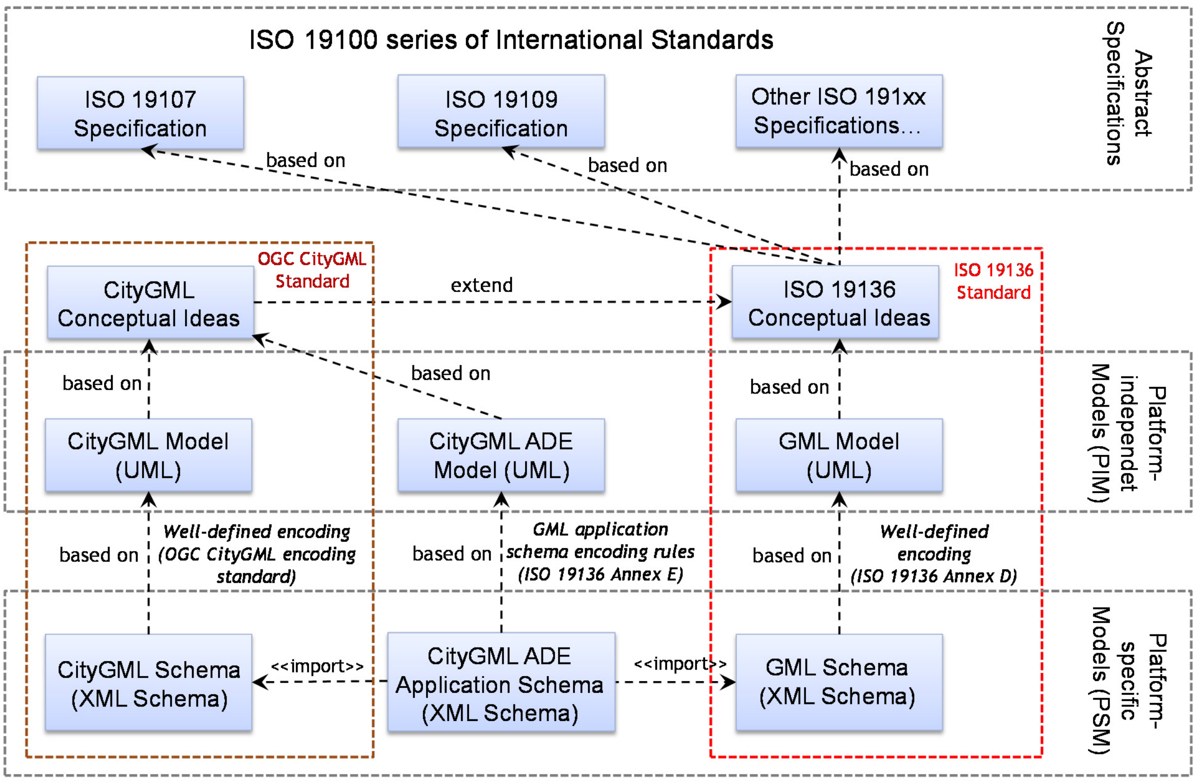
ISO 19100 series of International Standards
Abstract Specifications
ISO 19107 Specification
ISO 19109 Specification
Other ISO 191xx Specifications…
based on
based on
based on
OGC CityGML Standard
CityGML Conceptual Ideas
extend
ISO 19136 Standard
ISO 19136 Conceptual Ideas
based on
based on
based on
Platform-independent Models (PIM)
CityGML Model (UML)
CityGML ADE Model (UML)
GML Model (UML)
based on
Well-defined encoding (OGC CityGML encoding standard)
based on
GML application schema encoding rules (ISO 19136 Annex E)
based on
Well-defined encoding (ISO 19136 Annex D)
Platform-specific Models (PSM)
CityGML Schema (XML Schema)
<<import>>
CityGML ADE Application Schema (XML Schema)
<<import>>
GML Schema (XML Schema)

Fig. 2: Relationship between the ISO 19100 standard family and CityGML ADEs

As a GML application schema, CityGML specifies domain specific concepts for 3D city models formally by extending GML's data models for the spatio-semantic modelling of 3D city and landscape objects. It is a platform-independent geospatial information model that has been realized as a XML-based application schema (CityGML Schema) in compliance with GML schema.

According to the CityGML specification, CityGML is featured with an extension mechanism called "Application Domain Extension (ADE)" allowing to dynamically enrich the CityGML's predefined data models with extra classes and attribute properties to develop a domain-specific GML application schema. According to literature (VAN DEN BRINK et al. 2013), CityGML ADEs can also be originated and developed from the start with the design of a platform-independent model using UML models by following the MDA approach. An XML-based application schema pointing to the CityGML and GML schemas can later be automatically generated by means of the UML-to-XML encoding rules defined in the ISO 19136 standard (SHAPECHANGE 2016). Although the UML model can be directly serialized to a text-based document using the standard format "XML Metadata Interchange (XMI)" for information exchange, CityGML ADEs developed by third parties are mostly provided by means of XML schema definition (XSD) files instead. This is because, on the one hand, the XML schema provides a sophisticated syntactical structure which is able to fully represent the semantics of object-oriented models with complex data structures and can also be easily interpreted and parsed by many XML-schema-aware software tools and programming libraries. On the other hand, the XML instance documents can also be directly checked against their meta-models using the respective XML schemas to ensure the validity of the corresponding geospatial data.

3 Extendable relational database structure for CityGML ADEs

In 3D GIS applications, typically database management systems (DBMS) are employed for achieving high-efficient data storage and management as well as interoperable data access for large-volume geospatial data. Since the data models of CityGML ADEs may comprise heterogeneous information about the domain-specific features along with their spatial and non-spatial properties, a number of open-source and commercial database products with spatial extensions have been developed and also intensively investigated in research and development projects to ascertain their key functional capabilities and limitations in handling GML-compliant geospatial data. These database products can be roughly categorized into two types according to their native database structure, namely non-relational and relational databases. The former one can be further classified into object-oriented databases, document-oriented databases, and graph databases etc. According to the literature (MAO et al. 2014, AGOUB et al. 2016), such non-relational databases are currently still more or less limited in their capabilities and performance of performing certain kinds of spatial operations and coordinate transformations which are of great importance for the use of CityGML ADEs in practical applications. Thus, due to their extensive abilities for storing, analyzing, and processing spatial data elements, the spatially-enhanced relational database management systems (SRDBMS) such as the commercial Oracle with 'Spatial' license and the open-source PostgreSQL with 'PostGIS' extension etc. are nowadays predominantly employed in many enterprise applications and services to maintain GML-conformant data.
As a representative relational database solution, the open-source 3D geodatabase "3DCityDB" was developed for CityGML. It can be operated using the Oracle and or PostgreSQL database management system. The database schema of 3DCityDB results from a careful, manual mapping of the object-oriented data model of CityGML onto a compact relational database structure optimized with respect to database complexity, operating performance, and semantic

interoperability. Over the past years, the 3DCityDB has been widely deployed in many commercial production environments to manage virtual 3D models for many cities worldwide like Munich, Berlin, Zurich, Rotterdam, Helsinki, Singapore, and London etc. (KOLBE et al. 2016). However, the current version of the 3DCityDB does not provide a generic solution for handling datasets of CityGML ADEs. This limitation hinders the usage of 3DCityDB to be disseminated for many domain-specific application fields which in turn have a strong need for dynamically extending 3DCityDB for arbitrary CityGML ADEs. In order to achieve this objective, an extended database structure based on the original 3DCityDB database schema was proposed in the course of this research work and the conceptual database design is sketched in the following figure in terms of an informal package diagram.

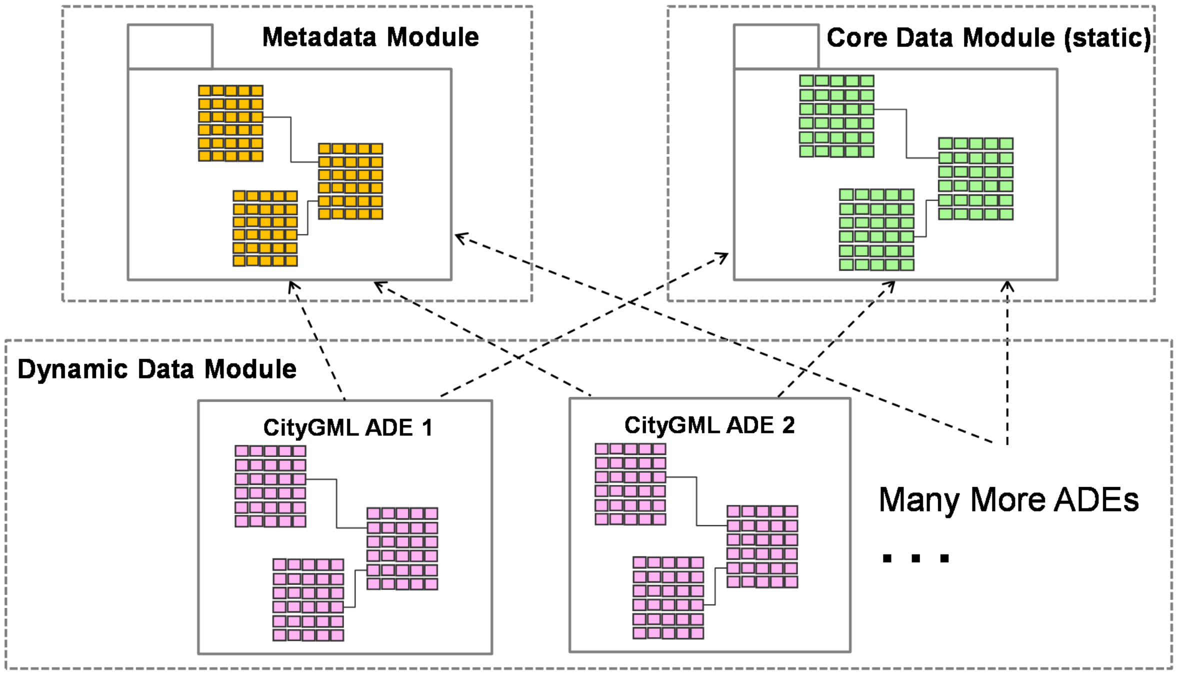

Fig. 3: Dynamically extendable database structure for storing CityGML ADE data in the 3DCityDB

As shown in the figure above, the new 3DCityDB database schema is designed in a modular fashion. It consists of three parts, namely *Metadata Module*, *Core Data Module*, and *Dynamic Data Module*. The green grids enclosed in the *Core Data Module* represents those database tables that are already included in the current version of the 3DCityDB database schema which is responsible for storing the standard CityGML models such as *Building*, *Tunnel, Transportation, CityFurniture, CityObjectGroup, Generic, Appearance* etc. For a given CityGML ADE, an additional group of database tables forming as a separate module belonging to the *Dynamic Data Module* (pink grids in the figure) shall be created and attached to the 3DCityDB database schema and the relationships (e.g. generalization/specialization and associations) among the model classes of CityGML and CityGML ADEs are adequately reflected using database foreign key constraints which can also ensure the data integrity and consistency within the database system. The *Metadata Module* associated with the *Dynamic Data Module* is used for storing the relevant meta-information (e.g. the XML namespaces, class affiliations etc.) about the application schema of the registered CityGML ADEs as well as about the database objects (e.g. indexes, columns, foreign key constrains) created in the corresponding *Dynamic Data Module*. In this way, the SQL statements for e.g. dropping the database tables of an individual CityGML ADE can be directly derived from the database schema which hence become to be dynamically manageable for handling multiple CityGML ADEs within the one database instance.

4 Automatic derivation of relational database schemas for CityGML ADEs using a Graph Transformation System

4.1 General concepts

Concerning the fact that domain-specific data models of CityGML ADEs may define arbitrarily complex data structures within their XML schema definition files, one of the key challenges regarding the use of the database structure presented in the previous section is to find a way to automatically derive efficient database schemas. Whereas the XML schema of a CityGML ADE natively represents an object-oriented data structure, the target database schema has a relational table structure which should result from a corresponding model transformation process to perform the mapping of an object-oriented model (input model) to a relational database model (output model). For this, both the input and output models have to be mapped onto some computer-interpretable representations such that the model transformation process can be automatically carried out by applying a set of predefined mapping rules within a computer-aided transformation engine. A couple of commercial and open-source software systems like Go Loader (SNOWFLAKE 2016) and Deegree (DEEGREE 2016) etc. have been developed for this problem which are capable of reading and parsing GML-compliant application schemas and automatically generating the desired relational database schemas for different types of modern database management systems such as Oracle and PostgreSQL/PostGIS.

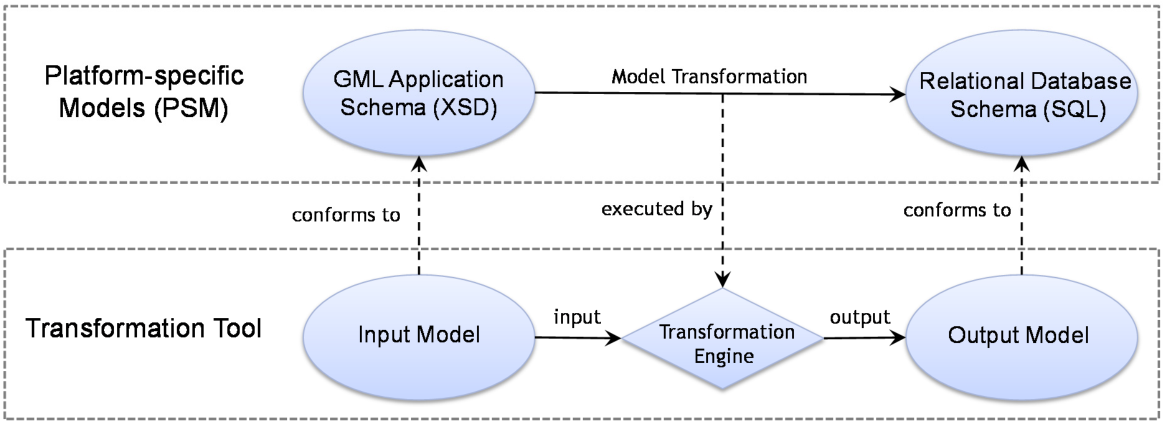

Fig. 4: General approach for deriving relational database schemas based on model transformation

However, these software systems are usually limited in performing such model transformation processes by following a number of simple and rigid mapping rules, i.e. each GML class or complex data type is mapped onto one individual database table, and an association between two GML classes is represented using an associative database table connecting with the database tables

of the respective GML classes. Such kinds of mapping rules can easily result in a large amount of database tables and will, therefore, lead to time-consuming data retrieval processes when, for example, performing a complex query on those data contents that are distributed over many database tables, because a large number of database joins are required which have a significant negative impact on the overall database performance. This issue has been taken into account by the software system "Go Loader" which provides a special optimization strategy that maps multiple associated feature classes onto one database table in order to ensure that the overall number of the generated database tables will not exceed a given maximum number. However, this solution lacks the ability to treat some special cases in a special way and can only rigidly reduce the database complexity without concerning the semantic clarity. For example, in case two non-abstract feature classes have an inheritance relationship and both are attached with different attribute properties, each feature class shall be mapped to an individual database table rather than a shared one, because such separated table representation is not only space-efficient for the database but also has much more clear class affiliation to facilitate the data access to the respective database tables for ETL software tools.
In order to achieve a good trade-off between database complexity and semantic clarity, the design decisions applied during the manual development of the 3DCityDB database schema must be captured and abstracted to a set of mapping rules which will be adopted in the generation process of relational database schemas for CityGML ADEs. The general idea of this design decision is that a group of classes with particular characteristics shall be first simplified into an optimal model structure which can be transformed to a relational database model with a simple mapping step (cf. STADLER et al. 2009). However, such design decision process strongly relies on many manual operations, because the class groups usually have varying characteristics and complex structures which are hardly formalized as programming code implemented in traditional software systems for performing automatic operations. Moreover, the mapping process can become even more complicated due to the possible iterative procedures in case that the simplified class groups need to be again further optimized if they satisfy some further simplification criteria. Concerning the above-mentioned issues, a novel approach is needed to overcome the above-mentioned issues with the help of an appropriate formalism to represent the source and target models as well as their mapping rules.

4.2 Mathematical background and conceptual solution

Since both the object-oriented model and the relational database model intrinsically have a graph structure, the semantic meanings and mapping relationships of both models can be represented using a special kind of graph whose nodes and edges shall be assigned with types and attributes. With such graph, the model transformation can be adequately abstracted as an algebraic graph transformation comprising a series of user-defined graph transformation rules $\{r_1,\ r_2, \dots, r_n\}$ that can be used to declaratively describe the model transformation rules and be successively applied using a graph transformation system (or graph rewriting system) for typed and attributed graphs. According to the fundamental theory of this system (EHRIG et al. 2004), each graph transformation rule is equivalent to a match morphism in the form of $r\colon L \rightarrow R$ where L is called left-hand side (LHS) graph, whereas R is called right-hand side (RHS) graph. The LHS graph can be seen as a match pattern which could be algebraically isomorphic to one of those graphs $\{G_1',\ G_2',\ \dots,\ G_n'\}$

that are a subset of the given host graph G_S, where $L \cong G'_x$ and $\{G'_1, G'_2, \ldots, G'_n\} \subseteq G_S$. Per default, the given transformation rules are ordered randomly by the graph transformation system and each will be selected and checked against the host graph G_S. Alternatively, the processing sequence can also be scheduled by grouping them into a set of layers sorted in descending order according to the user-defined priorities and as such will be processed successively. When a graph transformation rule is being processed and if a match of the LHS has been found in the host graph, this graph transformation rule will be interpreted as being applicable and the matched subgraph will be substituted by the RHS immediately. In the subsequent operation, the modified host graph G_i will in turn be treated as the input for the next transformation step and the entire graph transformation process will hence be carried out within an iterative procedure which will be terminated until no further applicable transformation rule can be found. In the last step, and the latest state of the host graph G_T will be treated as the final result of the graph transformation process (cf. Fig. 5).

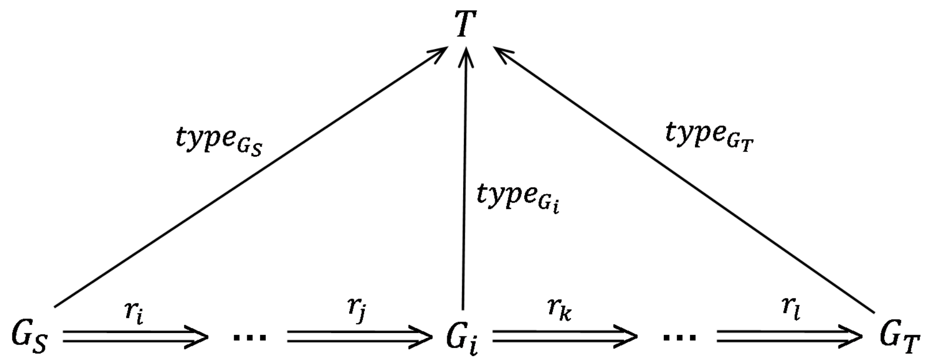
$$\begin{tikzcd}
 & & T & & \\
G_S \arrow[r, Rightarrow, "r_i"] \arrow[urr, "type_{G_S}"] & \cdots \arrow[r, Rightarrow, "r_j"] & G_i \arrow[u, "type_{G_i}"'] \arrow[r, Rightarrow, "r_k"] & \cdots \arrow[r, Rightarrow, "r_l"] & G_T \arrow[ull, "type_{G_T}"']
\end{tikzcd}$$

Fig. 5: General process chain in graph transformation systems (cf. TAENTZER et al. 2006)

In addition, a transformation rule can be further constrained by defining additional conditions which are generally categorized into two types: the attribute-based prerequisite condition called *Positive Application Condition* (PAC) and the graph-based prohibitive condition called *Negative Application Condition* (NAC). The PACs are combined with the logical operator “AND” and thus a graph transformation rule will only be triggered when all its PACs are fulfilled. In the contrary, the NACs use the logical operator “OR” and make a transformation rule not to be applicable if one of the NACs is satisfied. Therefore, the proper use of PACs and NACs offers a variety of possibilities for specifying a graph transformation rule being subject to different kinds of application conditions. For instance, in practical applications, there is a frequently applied NAC which is algebraically identical with the RHS of the respective transformation rule and can be used for avoiding the unexpected infinite loop of running the same graph transformation rule. Moreover, the graph transformation system supports the declaration of types for graph objects (e.g. nodes and edges) and also the definition of an abstracted meta-graph (or called *type graph*) T (cf. Fig. 5) which has a similar fashion compared to the UML meta-model in model-driven engineering and can serve as a global constraint on the declared node and edge types by prescribing the structural relationships among them, i.e. the associations and inheritance hierarchies between node types, the multiplicities and role names of the edges etc. Due to its rich semantic structure the meta-graph is well capable of formally describing the meta-models of the object-oriented models according to the ISO 19100 standard family as well as the underlying conceptual models of the relational database models along with the model mapping structures. In the following figure, an excerpt of our developed meta-graph is depicted in which the yellow nodes represent elements of the object-oriented (source) schema and the green nodes represent elements of the relational (target) database schema. Based on this graph representation, a model-driven framework can be established for the automatic derivation of a relational database schema from a GML application schema using graph transformation systems.

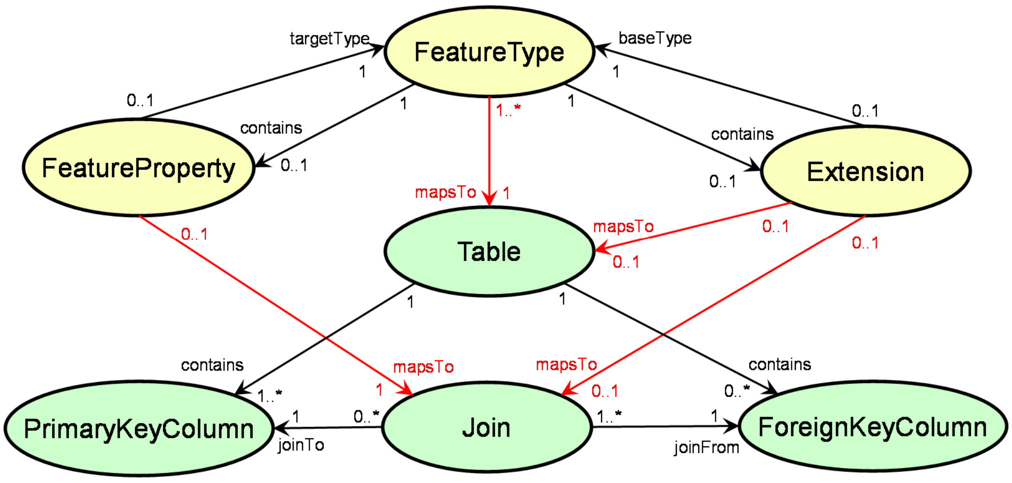

Fig. 6: An excerpt of the meta-graph for representing the conceptual model mapping structure

The yellow nodes together with the associated edges among them constitute a meta-subgraph for representing the structural information (e.g. associations and inheritance relationships) of those classes that are transitively derived from the GML class "FeatureType". The definition of this subgraph follows the XML encodings defined in the GML specification (ISO 19136) in accordance with the conceptual models of the ISO 19109 General Feature Model (GFM). First, the inheritance relationship is represented by means of the node type *Extension* being connected with the edge types *contains* and *baseType* both of which are connected with the node type *FeatureType*. This graph structure is adopted from the syntax of the extension mechanism of XML schema according to which a subclass defined as a XML <complexType> shall enclose an <extension> XML element having a reference to the respective super class. Second, the association such as aggregation and composition between two feature classes can be explicitly mapped onto a node with the type *FeatureProperty* which links the node type *FeatureType* via the edge types *contains* and *targetType*. This graph representation follows the so-called "feature-property-feature" encoding structure where an aggregate class represented as an XML element can contain a child element acting as a "property" element which shall have a reference to the member class in order to establish the class association.

The green nodes and their associated edges are meant to be used for semantically representing the meta-model of the relational database schema especially regarding the relations between database tables. In general, a database table shall have only one primary key acting as a unique record identifier which can be made up of one or more columns known as "primary key columns". In order to establish a connection between two database tables, one of them should have a special column called "foreign key column" pointing to a primary key column in another one by defining a so-called "foreign key constraint". It is also allowed to define multiple foreign key constraints on different columns to link with the one primary key column within the same table. Based on this conceptual database structure, a corresponding graph representation is designed in the following way: A node type *Table* representing the database table is created and connected with the node types *PrimaryKeyColumn* and *ForeignKeyColumn* to reflect the ownership using the edge type *contains*. The foreign key constraint is explicitly represented using the node type *Join* which links the *ForeignKeyColumn* with the *PrimaryKeyColumn* using the edge type *joinFrom* and *joinTo* respectively.

Concerning the conceptual models of the mapping relationships between object-oriented models and relational database models, the edges with the type *mapsTo* are utilized to link the corresponding entities of both models according to our following mapping principles:

- A feature class shall be mapped onto a database table which also allows to be shared by multiple feature classes if they have inheritance relationships among them.
- In case that two feature classes with inheritance relationship are mapped onto one database table, the inheritance relationship should also be mapped onto this database table. For other cases, an additional foreign key constraint shall be used for the representing the inheritance relationship.
- The association between two feature classes shall be always mapped onto a foreign key constraint.

Above, a variety of graph transformation rules can be flexibly designed by developers in a declarative way. In total, we have defined 11 of these rules to cover all occurring mappings. For the illustrative purpose, we have chosen a representative graph transformation rule realizing a relatively complex model transformation whose underlying idea is shown in the following figure.

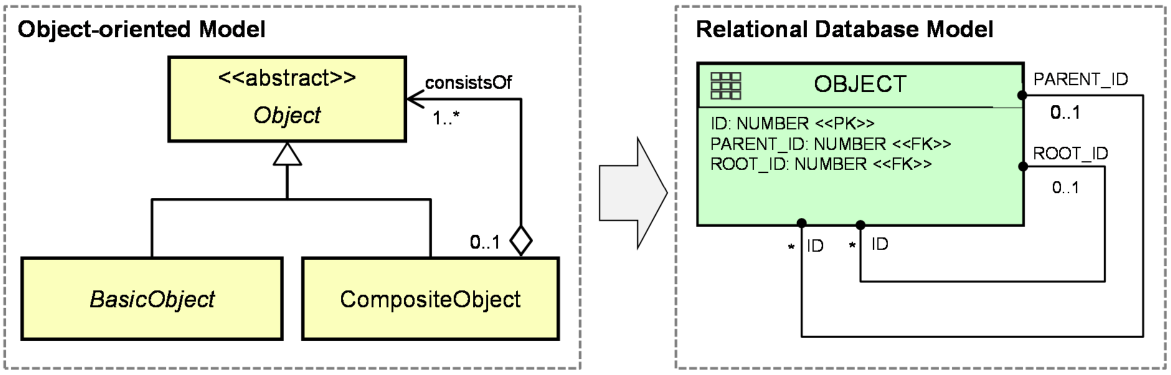

Fig. 7: General idea for the mapping of an object-oriented model with the composite pattern onto an efficient relational database model

The objected-oriented data model sketched on the left of the figure builds a recursively composite data structure according to the design pattern "Composite Pattern" (cf. GAMMA et al. 1995). The target database structure on the right side was originally introduced in the 3DCityDB documentation (KOLBE et al. 2016) and has been successfully implemented for the 3DCityDB database schema to perform fast and efficient queries on composite geospatial data stored in relational databases. The key idea of this database design is, on the one hand, to utilize one database table for the mapping of all the involved feature classes along with their inheritance relationships. On the other hand, a foreign key column PARENT_ID is used for representing the composition relationship. Additionally, this database table receives a foreign key column ROOT_ID which holds the ID of the root element of each composite hierarchy and hence allows for fast retrieving of all its child elements by querying on the attribute ROOT_ID in order to avoid time-costly recursive database joins. More details about the corresponding database implementation can be found in the technical documentation of 3DCityDB. The following figure gives an overview of how the presented mapping concept is formulated as a graph transformation rule with respect to the meta-graph illustrated in figure 6. The LHS graph shown on the left of figure 8 stands for the object-oriented model, whereas the RHS graph receives a copy of the LHS graph in addition with a set of new nodes and edges which represent the target database model and the explicit mapping information between the entities of both models. Moreover, this graph transformation rule can be additionally equipped with a NAC being structurally identical with the RHS in order to ensure that this transformation process will be executed only one time.

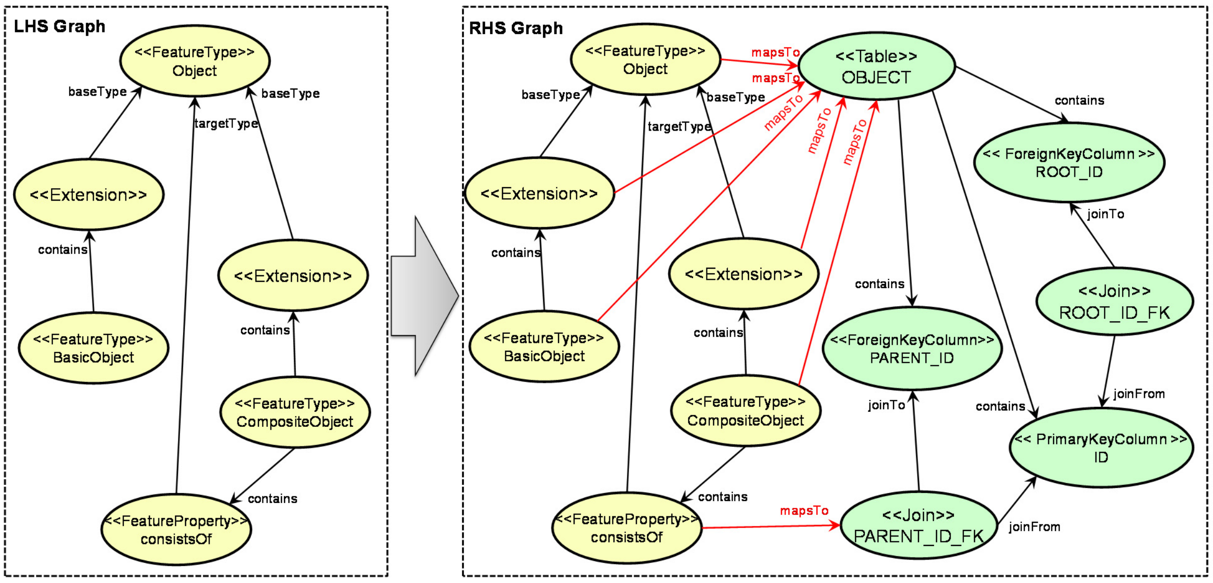

Fig. 8: Graph transformation rule in accordance with the mapping concept illustrated in Fig. 7

4.3 Prototypical implementation

This section introduces a converter tool which was implemented based on the presented conceptual solution presented in the previous sections regarding the automatic derivation of a 3DCityDB compliant relational database schema from the GML application schema of a CityGML ADE. This converter tool is as a stand-alone Java application which employs the AGG graph transformation system (TAENTZER et al. 2003) for performing the dedicated graph transformation process. For test purposes, a number of existing valid CityGML ADEs such as Energy ADE (NOUVEL et al. 2015), UtilityNetwork ADE (KUTZNER & KOLBE 2016), and Dynamizer ADE (CHATURVEDI & KOLBE 2016) have been used for automatically deriving their corresponding relational database schemas. These schemas are given as SQL scripts. When they are executed, new tables according to the CityGML ADEs are being created in the 3DCityDB. The tables extend to the latest version (3.3.0) of the 3DCityDB database schema and the SQL scripts are online available from a web-based GitHub repository[2]. The following figure gives an overview of the entire process chain of this converter tool which is logically decomposed into three parts and each is explained in detail in the following paragraph.

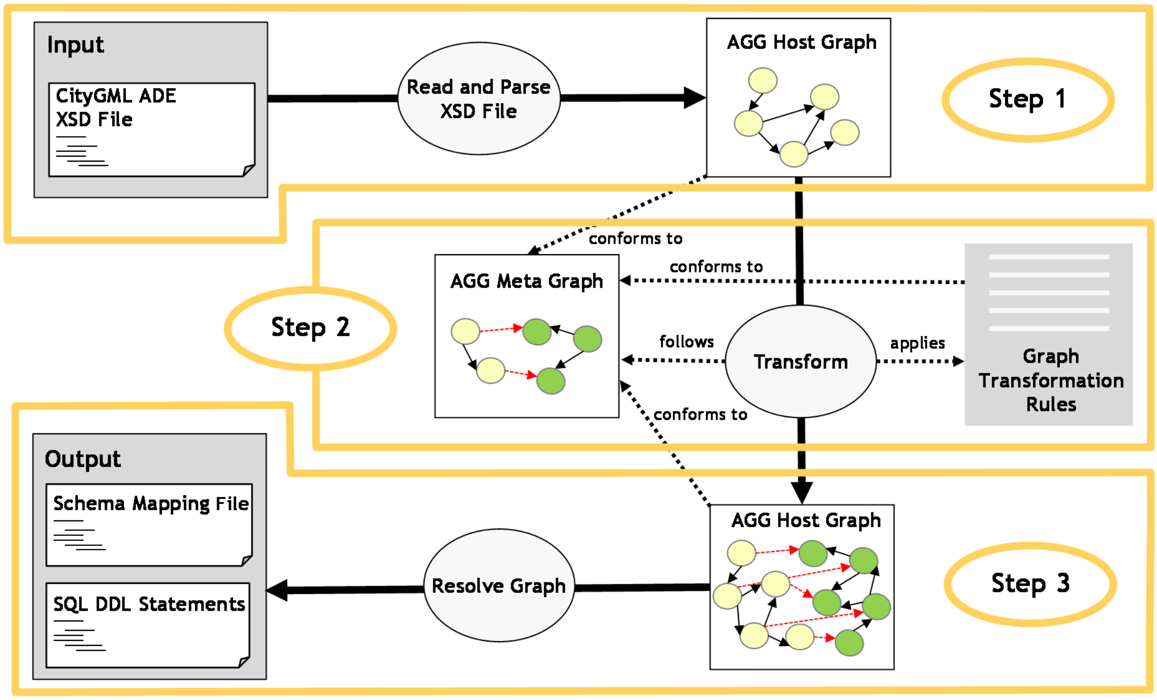

Fig. 9: Model Transformation using Graph Transformation System

In the first processing step, the XML schema definition (XSD) file of the input CityGML ADE is read into the converter tool using the Java library XSOM for reading and retrieving the underlying structural and semantic information from the XML elements. Subsequently, the parsed information is internally mapped onto Java objects forming an AGG-compliant graph representation of the object-oriented data model. In the next step, the created AGG graph will be used as the input graph for the AGG engine in order to apply the predefined graph transformation rules for performing the desired graph transformation process according to the mapping rules employed during the development of the 3DCityDB relational database schema. These graph transformation rules along with their supervisory meta-graph can be easily created by means of the graphical editor program delivered together with the AGG engine which allows for interactively defining and drawing the graph structures and saving them in XML format which can be reused by other AGG engines running on different platforms. After having completed the transformation process, the input graph together with the newly created nodes and edges will form the output graph representing the information of the object-oriented model and the derived relational database model as well as the mapping information between both modes. In the last step, the output graph will be resolved by the converter tool retrieving the target relational database schema which is then written to an SQL

[2] https://github.com/yaozhihang/3DCityDB-Extensions-for-CityGML-ADEs

document for a chosen database system like Oracle or PostgreSQL/PostGIS. Additionally, the mapping information can also be derived from the output graph in the form of an XML-based mapping schema that is used to facilitate the automatic data transformation between CityGML ADE instance documents and database systems via the standardized open interface WFS for web-based data access.

5 Conclusions and outlook

The 3D city models which are using CityGML together with its extension mechanism Application Domain Extension (ADE) are nowadays being broadly used as a common information hub for representing the most relevant urban objects along with their diverse graphical, topological, as well as semantic properties. As cities are highly complex systems in general whose underlying domain-specific information are usually very large in size and often also have complex data structures, the high-performance storage and management of such kinds of data information are usually required for a variety of simulations and analyses in the field of 3D GIS and pose a challenge to many researchers and software developers. The key intention of the presented work is to find and explore a new approach to dynamically extend the spatially-enhanced relational database management systems (SRDBMS) for efficiently handling the geospatial data elements of arbitrary CityGML ADEs.

A dynamically extendable database structure was first proposed which was designed based on the extension of the open source 3DCityDB database schema to provide an integrative database platform for handling multiple CityGML ADEs at the same time. Second, the automatic derivation of relational database schemas from the XML schema definition (XSD) files of CityGML ADEs was expressed as a special kind of model transformation process, in which the logical mapping rules between object-oriented models and relational database models were declaratively described as a set of user-defined graph transformation rules which can be carried out within a graph transformation system. Finally, based on the introduced conceptual solution, a Java-based desktop application has been developed to technically implement the automatic procedures ranging from the parsing and creation of a graph representation of the input CityGML ADE application schema via the execution of the dedicated graph transformation process up to the derivation of the target relational database schema in SQL format together with the respective mapping schema in terms of XML structure.

Future work will focus on the continuing development of the presented converter tool by improving and refining the existing graph transformation rules or adding new ones for automatically generating more compact and efficient relational database schemas. It is also planned to include this converter tool as well as the proposed extendable database structure into a future release of the 3DCityDB to provide the basis for handling arbitrary CityGML ADEs. For the practical use of this database solution, numerous challenging tasks are raised, including e.g. the development of a generic data access tool for interacting with the dynamically created database schema, since the current version of the 3DCityDB import/export tool doesn't support for reading and writing data elements of CityGML ADEs and hence must be conceptually as well as functionally extended. Moreover, an enhanced version of the WebGL-based 3D web client (Yao et al. 2016) being part of the 3DCityDB software package will be developed for the interactive 3D visualization and exploration of spatial and thematic information of any CityGML ADEs on the web. In summary, our long-term development goal is to make the 3DCityDB software package to be an extensive open-source application platform with full support for the interoperable management, access, analysis, and visualization of domain-extendable 3D city models according to the CityGML standard.

6 Acknowledgments

This work has been carried out in large parts within the research project "*Fachlich erweiterbare 3D-Stadtmodelle – Management, Visualisierung und Interaktion*" funded by the company CADFEM GmbH and its subsidiary company virtualcitySYSTEMS GmbH, whom we would like to thank for their support.

7 References

AGOUB, A., KUNDE, F. & KADA, M., 2016: Potential of Graph Databases in Representing and Enriching Standardized Geodata. Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V., Band **25**, T. KERSTEN (Hrsg.), 208-216.

CHATURVEDI, K. & KOLBE, T. H., 2016: Integrating Dynamic Data and Sensors with Semantic 3D City Models in the context of Smart Cities. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences **IV-2**/W1, 31-38.

DEEGREE, 2016: deegree webservices 3.3.19 documentation. https://wiki.snowflakesoftware.com/display/ GLD/XML%2C+GML+and+Application+Schemas, last access on 2016-12-13.

EHRIG, H., PRANGE, U. & TAENTZER, G., 2004: Fundamental theory for typed attributed graph transformation. International conference on graph transformation. Graph Transformations, Lecture Notes in Computer Science, EHRIG, H., ENGELS, G., PARISI-PRESICCE, F. & ROZENBERG, G. (eds.), Springer, Berlin Heidelberg, 161-177.

GRÖGER, G., KOLBE, T.H., NAGEL, C. & HÄFELE, K.-H., 2012: OGC City Geography Markup Language (CityGML) Encoding Standard Version 2.0.0. OGC Doc 12-019, Open Geospatial Consortium. Retrieved from http://www.opengeospatial.org/standards/citygml

GA, D., DJURIC, D. & DEVED, V., 2006: Model Driven Architecture and Ontology Development. Springer Science & Business Media.

GAMMA, E., HELM, R., JOHNSON, R. & VLISSIDES, J., 1995: Design Patterns: Elements of Reusable Object-oriented Software, Addison-Wesley, ISBN: 0-201-63361-2.

ISO 19107:2003: Geographic Information – Spatial schema (GML) (ISO 19107:2003). International Organization for Standardization

ISO 19109:2005: Geographic Information – Rules for application schemas (ISO 19109:2005). International Organization for Standardization

ISO 19136:2007: Geographic Information – Geography Markup Language (ISO 19136:2007). International Organization for Standardization

ISO 19118:2011: Geographic Information – Encoding (ISO 19118:2011). International Organization for Standardization

KUTZNER, T. & KOLBE, T. H., 2016: Extending Semantic 3D City Models by Supply and Disposal Networks for Analysing the Urban Supply Situation. Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V., Band **25**, T. KERSTEN (Hrsg.), 382-394.

KOLBE, T. H., 2009: Representing and exchanging 3D city models with CityGML. 3D geo-information sciences, Lecture Notes in Geoinformation and Cartography, LEE, J. & ZLATANOVA, S. (eds.), Springer, Berlin Heidelberg, 15-31.

KOLBE, T. H., YAO, Z., NAGEL, C., REDWEIK, R., WILLKOMM, P., HUDRA, G. & KUNDE, F., 2016: 3D City Database for CityGML Version 3.3.0. Documentation. Retrieved from http://www.3dcitydb.org/3dcitydb/documentation/

MAO, B., HARRIE, L., CAO, J., WU, Z., & SHEN, J., 2014: NoSQL Based 3D City Model Management System. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences **40**(4), 169.

NOUVEL, R., BAHU, J.M., KADEN, R., KAEMPF, J., CIPRIANO, P., LAUSTER, M., HAEFELE, K.H., MUNOZ, E., TOURNAIRE, O. & CASPER, E., 2015: Development of the CityGML Application Domain Extension Energy for Urban Energy Simulation. Proceedings of 14th international conference of the international building performance simulation association (ibpsa) 2015.

SHAPECHANGE, 2016: XML Schema – ShapeChange. http://shapechange.net/targets/xsd/, last access on 2016-12-13.

STADLER, A., NAGEL, C., KÖNIG, G. & KOLBE, T. H., 2009: Making interoperability persistent: A 3D geo database based on CityGML. 3D geo-information sciences, Lecture Notes in Geoinformation and Cartography, Lee, J. & Zlatanova, S. (eds.), Springer, Berlin Heidelberg, 175-192.

SNOWFLAKE, 2016: GO Loader Documentation – Snowflake Software Documentation Wiki. https://wiki.snowflakesoftware.com/display/GLD/Home, last access on 2016-12-13.

TAENTZER, G., 2003: AGG: A Tool Environment for Algebraic Graph Transformation. International Workshop on Applications of Graph Transformations with Industrial Relevance, Springer, Berlin Heidelberg, 481-488.

TAENTZER, G. & GARUGHI, G.T., 2006: A Graph-based Approach to Transform XML Documents. International Conference on Fundamental Approaches to Software Engineering, Springer, Berlin Heidelberg, 48-62.

VAN DEN BRINK, L., STOTER, J. & ZLATANOVA, S., 2013: UML-Based Approach to Developing a CityGML Application Domain Extension. Transactions in GIS **17**(6), 920-942.

YAO, Z., CHATURVEDI, C. & KOLBE, T.H., 2016: Browser-basierte Visualisierung großer 3D-Stadtmodelle durch Erweiterung des Cesium Web Globe. Geoinformationssysteme 2016.