Open Geospatial Consortium

Approval Date: 2012-03-09

Publication Date: 2012-04-04

External identifier of this OGC® document: http://www.opengis.net/spec/citygml/2.0

Reference number of this OGC® project document: OGC 12-019

Version: 2.0.0

Category: OpenGIS® Encoding Standard

Editors: Gerhard Gröger, Thomas H. Kolbe, Claus Nagel, Karl-Heinz Häfele

OGC City Geography Markup Language (CityGML) Encoding Standard

This document is an OGC Member approved international standard. This document is available on a royalty free, non-discriminatory basis. Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type:	OpenGIS® Encoding Standard
Document subtype:	Encoding
Document stage:	Approved for public release
Document language:	English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.
THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.
THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable

Contents

i. Abstract

CityGML is an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is an application schema for the Geography Markup Language version 3.1.1 (GML3), the extendible international standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211.

The aim of the development of CityGML is to reach a common definition of the basic entities, attributes, and relations of a 3D city model. This is especially important with respect to the cost-effective sustainable maintenance of 3D city models, allowing the reuse of the same data in different application fields.

ii. Preface and Acknowledgements

This is the official CityGML logo. For current news on CityGML and information about ongoing projects and fields of research in the area of CityGML see

http://www.citygml.org and http://www.citygmlwiki.org

OGC work on CityGML is discussed and coordinated by the OGC 3D Information Management (3DIM) Working Group.CityGMLwas initally implemented and evaluated as part of the OGC Web Services Testbed, Phase 4 (OWS-4) in the CAD/GIS/BIM thread.

Version 2.0 of this standards document was prepared by the OGC CityGML Standards Working Group (SWG). Future discussion and development will be led by the 3DIM Working Group.

For further information see http://www.opengeospatial.org/projects/groups/3dimwg

CityGML also continues to be developed by the members of the Special Interest Group 3D (SIG 3D) of the GDI-DE Geodateninfrastruktur Deutschland (Spatial Data Infrastructure Germany) in joint cooperation with the 3DIM Working Group and the CityGML SWG within OGC.

For further information see http://www.sig3d.org/

The preparation of the English document version and the European discussion has been supported by the European Spatial Data Research Organization (EuroSDR; formerly known as OEEPE) in an EuroSDR Commission III project.

For further information see http://www.eurosdr.net

iii. Submitting organizations

This International Standard was submitted to the Open Geospatial Consortium Inc. by the members of the CityGML 1.0 Standards Working Group of the OGC. Amongst others, this comprises the following organizations:

a) Autodesk, Inc. (primary submitter)

b) Bentley Systems, Inc. (primary submitter)

c) Technical University Berlin (submitter of technology)

d) Ordnance Survey, UK

e) University of Bonn, Germany

f) Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam

g) Institute for Applied Computer Science, Karlsruhe Institute of Technology

CityGML was originally developed by the Special Interest Group 3D (SIG 3D), 2002 – 2012 - www.citygml.org.

iv. Submission contact points

All questions regarding this document should be directed to the editors or the contributors (including participants in development, cf. clause v):

Name	Institution	Email
Prof. Dr. Thomas H. Kolbe Claus Nagel Alexandra Lorenz	Institute for Geodesy and Geoinformation Science, Technical University Berlin	thomas.kolbe<at>tu-berlin.de claus.nagel<at>tu-berlin.de alexandra.lorenz<at>tu-berlin.de
Dr. Gerhard Gröger Prof. Dr. Lutz Plümer Angela Czerwinski	Institute for Geodesy and Geoinformation, University of Bonn	Groeger<at>ikg.uni-bonn.de Pluemer<at>ikg.uni-bonn.de Czerwinski<at>ikg.uni-bonn.de
Haik Lorenz	Autodesk, Inc.	haik.lorenz<at>autodesk.com
Alain Lapierre Stefan Apfel Paul Scarponcini	Bentley Systems, Inc.	alain.lapierre<at>bentley.com stefan.apfel<at>bentley.com paul.scarponcini<at>bentley.com
Carsten Rönsdorf	Ordnance Survey, Great Britain	carsten.roensdorf<at>ordnancesurvey.co.uk
Prof. Dr. Jürgen Döllner	Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam	juergen.doellner<at>hpi.uni-potsdam.de
Dr. Joachim Benner Karl-Heinz Häfele	Institute for Applied Computer Science, Karlsruhe Institute of Technology	joachim.benner<at>kit.edu karl-heinz.haefele<at>kit.edu

v. Participants in development

Name	Institution
Ulrich Gruber, Sandra Schlüter	District Administration Recklinghausen, Cadastre Department, Germany
Frank Bildstein	Rheinmetall Defence Electronics, Germany
Rüdiger Drees	T-Systems Enterprise Services GmbH, Bonn, Germany
Andreas Kohlhaas	GIStec GmbH (formerly), Germany
Frank Thiemann	Institute for Cartography and Geoinformatics, University of Hannover
Martin Degen	Cadastre Department, City of Dortmund
Heinrich Geerling	Architekturbüro Geerling, Germany
Dr. Frank Knospe	Cadastre and Mapping Department, City of Essen,
Hardo Müller	Snowflake Software Ltd., Great Britain
Martin Rechner	rechner logistic, Germany
Jörg Haist Daniel Holweg	Fraunhofer Institute for Computer Graphics (IGD), Darmstadt, Germany
Prof. Dr. Peter A. Henning	Faculty for Computer Science, University of Applied Sciences, Karlsruhe, Germany
Rolf Wegener Stephan Heitmann	State Cadastre and Mapping Agency of North-Rhine Westphalia, Germany
Prof. Dr. Marc-O. Löwner	Institute for Geodesy and Photogrammetry, Technical University of Braunschweig
Dr. Egbert Casper	Zerna Ingenieure, Germany
Christian Dahmen	con terra GmbH, Germany
Nobuhiro Ishimaru	Hitachi, Ltd., Japan

Kishiko Maruyama Eiichiro Umino Takahiro Hirose	
Linda van den Brink	Geonovum, The Netherlands
Ron Lake David Burggraf	Galdos Systems Inc., Canada
Marie-Lise Vautier Emmanuel Devys	Institut géographique national, France
Mark Pendlington	Ordnance Survey, Great Britain

vi. Changes to the OGC® Abstract Specification

The OGC® Abstract Specification does not require changes to accommodate this OGC® standard.

vii. Acknowledgments

The SIG 3D wishes to thank the members of the CityGML Standards Working Group and the 3D Information Management (3DIM) Working Group of the OGC as well as all contributors of change requests and comments. In particular: Tim Case, Scott Simmons, Paul Cote, Clemens Portele, Jeffrey Bell, Chris Body, Greg Buehler, François Golay, John Herring, Jury Konga, Kai-Uwe Krause, Gavin Park, Richard Pearsall, George Percivall, Mauro Salvemini, Alessandro Triglia, David Wesloh, Tim Wilson, Greg Yetman, Jim Farley, Cliff Behrens, Lukas Herman, Danny Kita, and Simon Cox.

Further credits for careful reviewing and commenting of this document go to: Ludvig Emgard, Bettina Petzold, Dave Capstick, Mark Pendlington, Alain Lapierre, and Frank Steggink.

Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights. However, to date, no such rights have been claimed or identified.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Significant changes between CityGML version 2.0.0 and CityGML version 1.0.0 (OGC document no. 08-007r1):

- New thematic modules for the representation of tunnels and bridges;
- Additional boundary surfaces for the semantic classification of the outer shell of buildings and building parts (*OuterCeilingSurface* and *OuterFloorSurface*);
- LOD0 representation (footprint and roof egde representations) for buildings and building parts;
- Additional attributes denoting a city object's location with respect to the surrounding terrain and water surface (*relativeToTerrain* and *relativeToWater*);
- Additional generic attributes for measured values and attribute sets; and
- Redesign of the CityGML code list mechanism (enumerative attributes are now of type *gml:CodeType* which facilitates to provide additional *code lists* enumerating their possible attribute values).

Migration of existing CityGML 1.0 instances to valid 2.0 instances only requires changing the CityGML namespace and schema location values in the document to the actual 2.0 values.

0 Introduction

0.1 Motivation

An increasing number of cities and companies are building virtual 3D city models for different application areas like urban planning, mobile telecommunication, disaster management, 3D cadastre, tourism, vehicle and pedestrian navigation, facility management and environmental simulations. Furthermore, in the implementation of the European Environmental Noise Directive (END, 2002/49/EC) 3D geoinformation and 3D city models play an important role.

In recent years, most virtual 3D city models have been defined as purely graphical or geometrical models, neglecting the semantic and topological aspects. Thus, these models could almost only be used for visualisation purposes but not for thematic queries, analysis tasks, or spatial data mining. Since the limited reusability of models inhibits the broader use of 3D city models, a more general modelling approach had to be taken in order to satisfy the information needs of the various application fields.

CityGML is a common semantic information model for the representation of 3D urban objects that can be shared over different applications. The latter capability is especially important with respect to the cost-effective sustainable maintenance of 3D city models, allowing the possibility of selling the same data to customers from different application fields. The targeted application areas explicitly include city planning, architectural design, tourist and leisure activities, environmental simulation, mobile telecommunication, disaster management, homeland security, real estate management, vehicle and pedestrian navigation, and training simulators.

CityGML is designed as an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is implemented as an application schema of the Geography Markup Language 3 (GML3), the extendible international standard for spatial data exchange and encoding issued by the Open Geospatial Consortium (OGC) and the ISO TC211. CityGML is based on a number of standards from the ISO 191xx family, the Open Geospatial Consortium, the W3C Consortium, the Web 3D Consortium, and OASIS.

CityGML defines the classes and relations for the most relevant topographic objects in cities and regional models with respect to their geometrical, topological, semantical, and appearance properties. "City" is broadly defined to comprise not just built structures, but also elevation, vegetation, water bodies, "city furniture", and more. Included are generalisation hierarchies between thematic classes, aggregations, relations between objects, and spatial properties. CityGML is applicable for large areas and small regions and can represent the terrain and 3D objects in different levels of detail simultaneously. Since either simple, single scale models without topology and few semantics or very complex multi-scale models with full topology and fine-grained semantical differentiations can be represented, CityGML enables lossless information exchange between different GI systems and users.

0.2 Historical background

CityGML has been developed since 2002 by the members of the Special Interest Group 3D (SIG 3D). Since 2010, this group is part of the initiative Spatial Data Infrastructure Germany (GDI-DE). Before 2010, the SIG 3D was affiliated to the initiative Geodata Infrastructure North Rhine-Westphalia (GDI NRW). The SIG 3D is an open group consisting of more than 70 companies, municipalities, and research institutions from Germany, Great Britain, Switzerland, and Austria working on the development and commercial exploitation of interoperable 3D city models and geovisualisation. Another result of the work from the SIG 3D is the proposition of the Web 3D Service (W3DS), a 3D portrayal service that is also being discussed in the Open Geospatial Consortium (OGC Doc. No. 05-019 and OGC Doc. No. 09-104r1).

A first successful implementation and evaluation of a subset of CityGML has been performed in the project "Pilot 3D" of the GDI NRW in 2005. Participants came from all over Germany and demonstrated city planning scenarios and tourist applications. By the beginning of 2006, a CityGML project within EuroSDR (European Spatial Data Research) started focusing on the European harmonisation of 3D city modelling. From June to December 2006, CityGML was employed and evaluated in the CAD/GIS/BIM thread of the OpenGIS Web Services Testbed #4 (OWS-4). Since 2008, CityGML (version 1.0.0) is an adopted OGC standard.

From that point in time, CityGML has disseminated worldwide. Many cities in Germany and in other countries in Europe provide their 3D city model in CityGML (Berlin, Cologne, Dresden and Munich, to mention only a few). In France, the project Bâti3D (IGN France) defines a profile of CityGML LOD2 and provides data from Paris and the city centres of Aix-en-Provence, Lille, Nantes and Marseille. CityGML also plays an important role in the pilot 3D project to obtain a 3D geoinformation standard and a 3D infrastructure for The Netherlands. Many cities in Europe like Monaco, Geneva, Zurich, Leewarden use CityGML LOD 2 or 3 to represent and exchange data, as well as cities in Denmark (LOD 2 and 3, partly LOD4). CityGML has strongly influenced the building model (version 2.0) of the INSPIRE initiative of the EU commission, which aims at the creation of an European spatial data infrastructure providing public sector data in an interoperable way. In Asia, the 3D city models of Istanbul (LOD 1 and 2), Doha, Katar (LOD3), and Yokohama (LOD2) are represented and exchanged in CityGML. Moreover, CityGML plays a crucial role for the 3D Spatial data infrastructure in Malaysia.

Today many commercial and academic tools support CityGML by providing import interfaces, export interfaces or both. An example is the 3D City Database which is a free and open source 3D geo database to store, represent, and manage virtual 3D city models on top of Oracle 10g R2 and 11g R1/R2 provided by the Technische Universität Berlin. It fully supports CityGML and is shipped with a tool for the import and export of CityGML models. Furthermore, an open source Java class library and API for the processing of CityGML models (citygml4j) is provided by the Technische Universität Berlin. The conversion tool FME (Feature Manipulation Engine) from Safe Software Inc., which is part of the interoperability extension of ESRI's ArcGIS, has read and write interfaces for CityGML. The same applies to CAD tools as BentleyMap from Bentley Systems as well as to GIS tools like SupportGIS from CPA Geo-Information. Many 3D viewers (which all are freely available) provide read interfaces for CityGML: the Aristoteles Viewer from the University of Bonn, LandXplorer CityGML Viewer from Autodesk Inc. (the studio version for authoring and management is not free) and the FZKViewer for IFC and CityGML from KIT Karlsruhe and BS Contact from Bitmanagement Software GmbH which offers a CityGML plugin for the geospatial extension BS Contact Geo. This enumeration of software tools is not exhaustive and steadily growing. Please refer to the official website of CityGML at http://www.citygml.org as well as the CityGML Wiki at http://www.citygmlwiki.org for more information.

0.3 Additions in CityGML 2.0

CityGML 2.0 is a major revision of the previous version 1.0 of this International Standard (OGC Doc. No. 08-007r1), and introduces substantial additions and new features to the thematic model of CityGML. The revision was originally planned to be a minor update to version 1.1. The main endeavor of the revision process was to ensure backwards compatibility both on the level of the conceptual model and on the level of CityGML instance documents. However, some changes could not be implemented consistent with directives for minor revisions and backwards compatibility as enforced by OGC policy (cf. OGC Doc. No. 135r11). The major version number change to 2.0 is therefore a consequence of conforming to the OGC versioning policy without having to abandon any changes or additions which reflect requests from the CityGML community.

CityGML 2.0 is backwards compatible with version 1.0 in the following sense: each valid 1.0 instance is a valid 2.0 instance provided that the CityGML namespaces and schema locations in the document are changed to their actual 2.0 values. This step is required because the CityGML version number is encoded in these values, but no further actions have to be taken. Hence, there is a simple migration path from existing CityGML 1.0 instances to valid 2.0 instances.

The following clauses provide an overview of what is new in CityGML 2.0.

New thematic modules for the representation of bridges and tunnels

Bridges and tunnels are important objects in city and landscape models. They are an essential part of the transportation infrastructure and are often easily recognizable landmarks of a city. CityGML 1.0 has been lacking thematic modules dedicated to bridges and tunnels, and thus such objects had to be modelled and exchanged using a *GenericCityObject* as proxy (cf. chapter 10.12). CityGML 2.0 now introduces two new thematic modules for the explicit representation of bridges and tunnels which complement the thematic model of CityGML: the *Bridge* module (cf. chapter 10.4) and the *Tunnel* module (cf. chapter 10.5).

Bridges and tunnels can be represented in LOD 1 – 4 and the underlying data models have a coherent structure with the *Building* model. For example, bridges and tunnels can be decomposed into parts, thematic boundary surfaces with openings are available to semantically classify parts of the shell, and installations as well as interi-

or built structures can be represented. This coherent model structure facilitates the similar understanding of semantic entities and helps to reduce software implementation efforts. Both the *Bridge* and the *Tunnel* model introduce further concepts and model elements which are specific to bridges and tunnels respectively.

Additions to existing thematic modules

- *CityGML Core module (cf. chapter 10.1)*
  Two new optional attributes have been added to the abstract base class *core:_CityObject* within the *CityGML Core* module: *relativeToTerrain* and *relativeToWater*. These attributes denote the feature's location with respect to the terrain and water surface in a qualitative way, and thus facilitate simple and efficient queries (e.g., for the number of subsurface buildings) without the need for an additional digital terrain model or a model of the water body.
- *Building module (cf. chapter 10.3)*
  - *LOD0 representation*
    Buildings can now be represented in LOD0 by footprint and/or roof edge polygons. This allows the easy integration of existing 2D data and of roof reconstructions from aerial and satellite imagery into a 3D city model. The representations are restricted to horizontal, 3-dimensional surfaces.
  - *Additional thematic boundary surfaces*
    In order to semantically classify parts of the outer building shell which are neither horizontal wall surfaces nor parts of the roof, two additional boundary surfaces are introduced: *OuterFloorSurface* and *OuterCeilingSurface*.
  - *Additional relations to thematic boundary surfaces*
    In addition to *_AbstractBuilding* and *Room*, the surface geometries of *BuildingInstallation* and *IntBuildingInstallation* features can now be semantically classified using thematic boundary surfaces. For example, this facilitates the semantic differentiation between roof and wall surfaces of dormers which are modeled as *BuildingInstallation*.
  - *Additional use of implicit geometries*
    Implicit geometries (cf. chapter 8.3) are now available for the representation of *_Opening*, *BuildingInstallation*, and *IntBuildingInstallation* in addition to *BuildingFurniture*. A prototypical geometry for these city objects can thus be stored once and instantiated at different locations in the 3D city model.
- *Generics module (cf. chapter 10.12)*
  Two generic attributes have been added to the *Generics* module: *MeasureAttribute* and *GenericAttributeSet*. A *MeasureAttribute* facilitates the representation of measured values together with a reference to the employed unit. A *GenericAttributeSet* is a named collection of arbitrary generic attributes. It provides an optional *codeSpace* attribute to denote the authority organization who defined the attribute set.
- *LandUse module (cf. chapter 10.10)*
  The scope of the feature type *LandUse* has been broadened to comprise both areas of the earth's surface dedicated to a specific land use and areas of the earth's surface having a specific land cover with or without vegetation.
- *Attributes* class, function, *and* usage *(all modules)*
  In order to harmonize the use of the attributes *class*, *function*, and *usage*, this attribute triplet has been complemented for all feature classes that at least provided one of the attributes in CityGML 1.0.

Additions to the CityGML code list mechanism

In CityGML, code lists providing the allowed values for enumerative attributes such as *class*, *function*, and *usage* can be specified outside the CityGML schema by any organization or information community according to their specific information needs. This mechanism is, however, not fully reflected in the CityGML 1.0 encoding schema, because in a CityGML 1.0 instance document a corresponding attribute cannot point to the dictionary with the used code list values. This has been corrected for CityGML 2.0: All attributes taking values from code lists are now of type *gml:CodeType* following the GML 3.1.1 mechanism for the encoding of code list values (cf.

chapter 10.14 for more information). The *gml:CodeType* adds an optional *codeSpace* value to enumerative attributes which allows for providing a persistent URI pointing to the corresponding dictionary.

Changelog for CityGML 2.0

Changes on the level of XML schema components are provided in Annex F.

Further edits to the specification document

- *Accuracy requirements for Levels of Detail (LOD) (cf. chapter 6.2)*
  The accuracy requirements for the different CityGML LODs proposed in chapter 6.2 are non-normative. The wording of chapter 6.2 in CityGML 1.0 is however inconsistent with regard to this fact and thus has been clarified for CityGML 2.0.
- *Rework of the CityGML example datasets (cf. Annex G)*
  The CityGML examples provided in Annex G have been reworked and extended. They now show a consistent building model in all five LODs and demonstrate, for example, the semantic and geometric refinement of the building throughout the different LODs as well as the usage of XLinks to share geometry elements between features. The datasets are shipped with the CityGML XML Schema package, and are available at http://schemas.opengis.net/citygml/examples/2.0/.
- *New example for the usage of Application Domain Extensions (cf. Annex I)*
  A second example for the usage of Application Domain Extensions in the field of Ubiquitous Network Robots Services has been added in Annex I.

OGC® City Geography Markup Language (CityGML) Encoding Standard

1 Scope

This document is an OGC Encoding Standard for the representation, storage and exchange of virtual 3D city and landscape models. CityGML is implemented as an application schema of the Geography Markup Language version 3.1.1 (GML3).

CityGML models both complex and georeferenced 3D vector data along with the semantics associated with the data. In contrast to other 3D vector formats, CityGML is based on a rich, general purpose information model in addition to geometry and appearance information. For specific domain areas, CityGML also provides an extension mechanism to enrich the data with identifiable features under preservation of semantic interoperability.

Targeted application areas explicitly include urban and landscape planning; architectural design; tourist and leisure activities; 3D cadastres; environmental simulations; mobile telecommunications; disaster management; homeland security; vehicle and pedestrian navigation; training simulators and mobile robotics.

CityGML is considered a source format for 3D portraying. The semantic information contained in the model can be used in the styling process which generates computer graphics represented e.g. as KML/COLLADA or X3D files. The appropriate OGC Portrayal Web Service for this process is the OGC Web 3D Service (W3DS). An image-based 3D portrayal service for virtual 3D landscape and city models is provided by the OGC Web View Service (WVS).

Features of CityGML:

- Geospatial information model (ontology) for urban landscapes based on the ISO 191xx family
- GML3 representation of 3D geometries, based on the ISO 19107 model
- Representation of object surface characteristics (e.g. textures, materials)
- Taxonomies and aggregations
    - Digital Terrain Models as a combination of (including nested) triangulated irregular networks (TINs), regular rasters, break and skeleton lines, mass points
    - Sites (currently buildings, bridges, and tunnels)
    - Vegetation (areas, volumes, and solitary objects with vegetation classification)
    - Water bodies (volumes, surfaces)
    - Transportation facilities (both graph structures and 3D surface data)
    - Land use (representation of areas of the earth's surface dedicated to a specific land use)
    - City furniture
    - Generic city objects and attributes
    - User-definable (recursive) grouping
- Multiscale model with 5 well-defined consecutive Levels of Detail (LOD):
    - LOD0 – regional, landscape
    - LOD1 – city, region
    - LOD2 – city districts, projects
    - LOD3 – architectural models (outside), landmarks
    - LOD4 – architectural models (interior)
- Multiple representations in different LODs simultaneously; generalisation relations between objects in different LODs
- Optional topological connections between feature (sub)geometries
- Application Domain Extensions (ADE): Specific "hooks" in the CityGML schema allow to define application specific extensions, for example for noise pollution simulation, or to augment CityGML by properties of the new National Building Information Model Standard (NBIMS) in the U.S.

2 Conformance

Conformance targets addressed by this International standard are CityGML instance documents only. Future revisions of this International Standard may also address consumers or producers as conformance targets.

Clauses 8 to 10 of this International standard specify separate CityGML XML Schema definitions and normative aspects, i.e. CityGML modules, which shall be used in CityGML instance documents in accordance with clause 7. Implementations are not required to support the full range of capabilities provided by the universe of all CityGML modules. Valid partial implementations are supported following the rules and guidelines for CityGML profiles in chapter 7.2.

CityGML instance documents claiming conformance to this International Standard shall:

a) conform to the rules and requirements specified in clauses 7 to 10;

b) pass all relevant test cases of the abstract test suite in annex B.1;

c) satisfy all relevant conformance classes of the abstract test suite related to CityGML modules in annex B.2.

3 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of OGC 12-019. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of OGC 12-019 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

The following documents are indispensable for the application of the CityGML standard. The geometry model of GML 3.1.1 is used except for some added concepts like implicit geometries (cf. chapter 8.2). The appearance model (cf. chapter 9) draws concepts from both *X3D* and *COLLADA*. Addresses are represented using the OASIS extensible Address Language *xAL*.

ISO 8601:2004, *Data elements and interchange formats – Information interchange – Representation of dates and times*

ISO/TS 19103:2005, *Geographic Information – Conceptual Schema Language*

ISO 19105:2000, *Geographic information – Conformance and testing*

ISO 19107:2003, *Geographic Information – Spatial Schema*

ISO 19109:2005, *Geographic Information – Rules for Application Schemas*

ISO 19111:2003, *Geographic information – Spatial referencing by coordinates*

ISO 19115:2003, *Geographic Information – Metadata*

ISO 19123:2005, *Geographic Information – Coverages*

ISO/TS 19139:2007, *Geographic Information – Metadata – XML schema implementation*

ISO/IEC 19775:2004, *X3D Abstract Specification*

OpenGIS® Abstract Specification Topic 0, *Overview, OGC document 04-084*

OpenGIS® Abstract Specification Topic 5, *The OpenGIS Feature, OGC document 99-105r2*

OpenGIS® Abstract Specification Topic 8, *Relations between Features, OGC document 99-108r2*

OpenGIS® Abstract Specification Topic 10, *Feature Collections, OGC document 99-110*

OpenGIS® Geography Markup Language Implementation Specification, *Version 3.1.1, OGC document 03-105r1*

OpenGIS® GML 3.1.1 Simple Dictionary Profile, *Version 1.0.0, OGC document 05-099r2*

IETF RFC 2045 & 2046, *Multipurpose Internet Mail Extensions (MIME). (November 1996)*

IETF RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax. (August 1998)*

W3C XLink, *XML Linking Language (XLink) Version 1.0. W3C Recommendation (27 June 2001)*

W3C XMLName, *Namespaces in XML. W3C Recommendation (14 January 1999)*

W3C XMLSchema-1, *XML Schema Part 1: Structures. W3C Recommendation (2 May 2001)*

W3C XMLSchema-2, *XML Schema Part 2: Datatypes. W3C Recommendation (2 May 2001)*

W3C XPointer, *XML Pointer Language (XPointer) Version 1.0. W3C Working Draft (16 August 2002)*

W3C XML Base, *XML Base, W3C Recommendation (27 June 2001)*

W3C XML, *Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation (6 October 2000)*

OASIS (Organization for the Advancement of Structured Information Standards): extensible Address Language (xAL v2.0).

Khronos Group Inc.: COLLADA – Digital Asset Schema Release 1.4.1

The Schematron Assertion Language 1.5. Rick Jelliffe 2002-10-01

4 Conventions

4.1 Abbreviated terms

The following abbreviated terms are used in this document:

2D	Two Dimensional
3D	Three Dimensional
AEC	Architecture, Engineering, Construction
ALKIS	German National Standard for Cadastral Information
ATKIS	German National Standard for Topographic and Cartographic Information
B-Rep	Boundary Representation
bSI	buildingSMART International
CAD	Computer Aided Design
COLLADA	Collaborative Design Activity
CSG	Constructive Solid Geometry
DTM	Digital Terrain Model
DXF	Drawing Exchange Format
EuroSDR	European Spatial Data Research Organisation
ESRI	Environmental Systems Research Institute
FM	Facility Management
GDF	Geographic Data Files
GDI-DE	Spatial Data Infrastructure Germany (Geodateninfrastruktur Deutschland)
GDI NRW	Geodata Infrastructure North-Rhine Westphalia
GML	Geography Markup Language
IAI	International Alliance for Interoperability (now buildingSMART International (bSI))
IETF	Internet Engineering Task Force
IFC	Industry Foundation Classes
ISO	International Organization for Standardisation
LOD	Level of Detail
NBIMS	National Building Information Model Standard
OASIS	Organisation for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium
OSCRE	Open Standards Consortium for Real Estate

SIG 3D	Special Interest Group 3D of the GDI-DE
TC211	ISO Technical Committee 211
TIC	Terrain Intersection Curve
TIN	Triangulated Irregular Network
UML	Unified Modeling Language
URI	Uniform Resource Identifier
VRML	Virtual Reality Modeling Language
W3C	World Wide Web Consortium
W3DS	OGC Web 3D Service
WFS	OGC Web Feature Service
X3D	Open Standards XML-enabled 3D file format of the Web 3D Consortium
XML	Extensible Markup Language
xAL	OASIS extensible Address Language

4.2 UML Notation

The CityGML standard is presented in this document in diagrams using the Unified Modeling Language (UML) static structure diagram (see Booch et al. 1997). The UML notations used in this standard are described in the diagram below (Fig. 1).

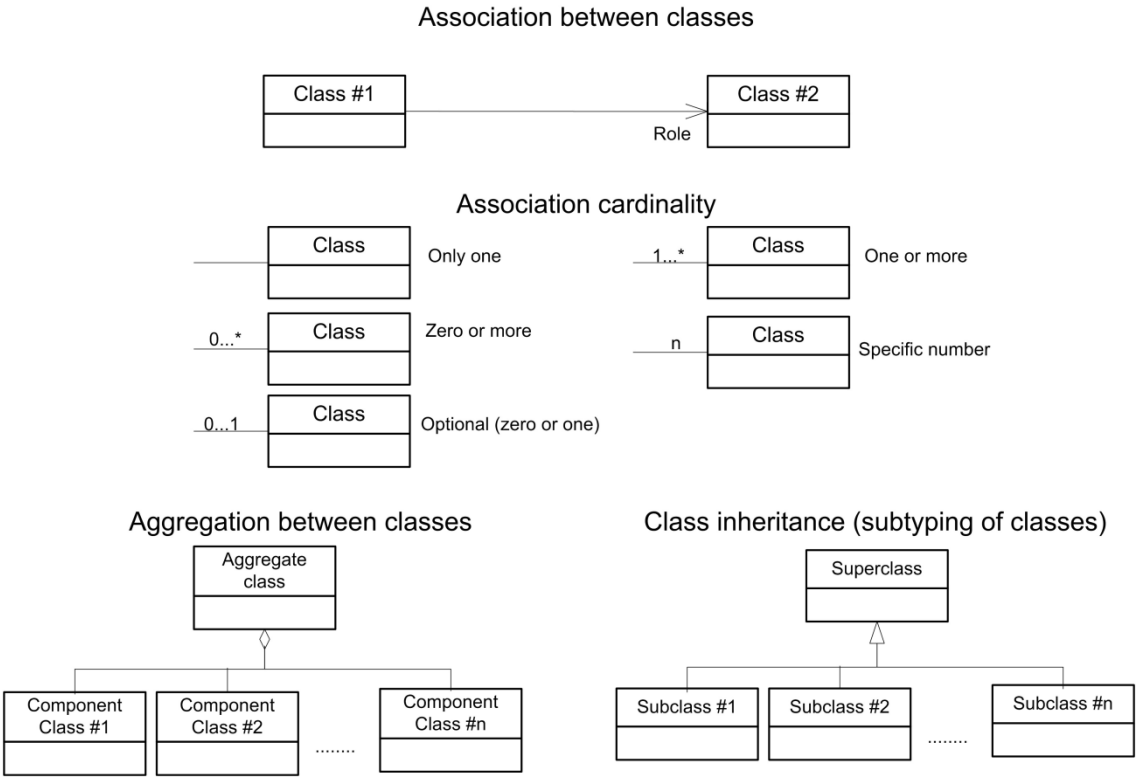

Fig. 1: UML notation (see ISO TS 19103, Geographic information - Conceptual schema language).

According to GML3 all associations between model elements in CityGML are uni-directional. Thus, associations in CityGML are navigable in only one direction. The direction of navigation is depicted by an arrowhead. In general, the context an element takes within the association is indicated by its role. The role is displayed near the target of the association. If the graphical representation is ambiguous though, the position of the role has to be drawn to the element the association points to.

The following stereotypes are used:

<<Geometry>> represents the geometry of an object. The geometry is an identifiable and distinguishable object that is derived from the abstract GML type *AbstractGeometryType*.

<<Feature>> represents a thematic feature according to the definition in ISO 19109. A feature is an identifiable and distinguishable object that is derived from the abstract GML type *AbstractFeatureType*.

<<Object>> represents an identifiable and distinguishable object that is derived from the abstract GML type *AbstractGMLType*.

<<Enumeration>> enumerates the valid attribute values in a fixed list of named literal values. Enumerations are specified inline the CityGML schema.

<<CodeList>> enumerates the valid attribute values. In contrast to Enumeration, the list of values is open and, thus, not given inline the CityGML schema. The allowed values can be provided within an external code list. It is recommended that code lists are implemented as simple dictionaries following the GML 3.1.1 Simple Dictionary Profile (cf. chapter 6.6 and chapter 10.14).

<<Union>> is a list of attributes. The semantics are that only one of the attributes can be present at any time.

<<PrimitiveType>> is used for representations supported by a primitive type in the implementation.

<<DataType>> is used as a descriptor of a set of values that lack identity. Data types include primitive predefined types and user-definable types. A DataType is thus a class with few or no operations whose primary purpose is to hold the abstract state of another class for transmittal, storage, encoding or persistent storage.

<<Leaf>> is used within UML package diagrams to indicate model elements that can have no further subtypes.

<<XSDSchema>> is used within UML package diagrams to denote the root element of an XSD Schema containing all the definitions for a particular namespace. All the package contents or component classes are placed within the one schema.

<<ApplicationSchema>> is used within UML package diagrams to denote an XML Schema definition fundamentally dependent on the concepts of another independent Standard within the XML Schema metalanguage. For example, ApplicationSchema indicates extensions of GML consistent with the GML "rules for application schemas".

In order to enhance the readability of the CityGML UML diagrams, classes are depicted in different colors if they belong to different UML packages (see Fig. 8 for an overview of UML packages). The following coloring scheme is applied:

- Classes painted in yellow belong to the UML package which is subject of discussion in that clause of the specification in which the UML diagram is given. For example, in the context of chapter 10.1 which introduces the *CityGML Core* module, the yellow color is used to denote classes which are defined in the *CityGML Core* UML package. Likewise, the yellow classes shown in UML diagrams in chapter 10.3 are associated with the *Building* module which is subject of discussion in that chapter.

- Classes painted in blue belong to a CityGML UML package different to that associated with the yellow color. In order to explicitly denote the UML package of such classes, their class names carry a namespace prefix which is uniquely associated with a CityGML module throughout this specification (cf. section 4.3 for a list of namespaces and prefixes). For example, in the context of the *Building* module, classes from the *CityGML Core* module are painted in blue and their class names are preceded by the prefix *core*.

- Classes painted in green are defined in GML3 and their class names are preceded by the prefix *gml*.

The following example UML diagram demonstrates the UML notation and coloring scheme used throughout this specification. In this example, the yellow classes are associated with the CityGML *Building* module, the blue classes are from the *CityGML Core* module, and the green class depicts a geometry element defined by GML3.

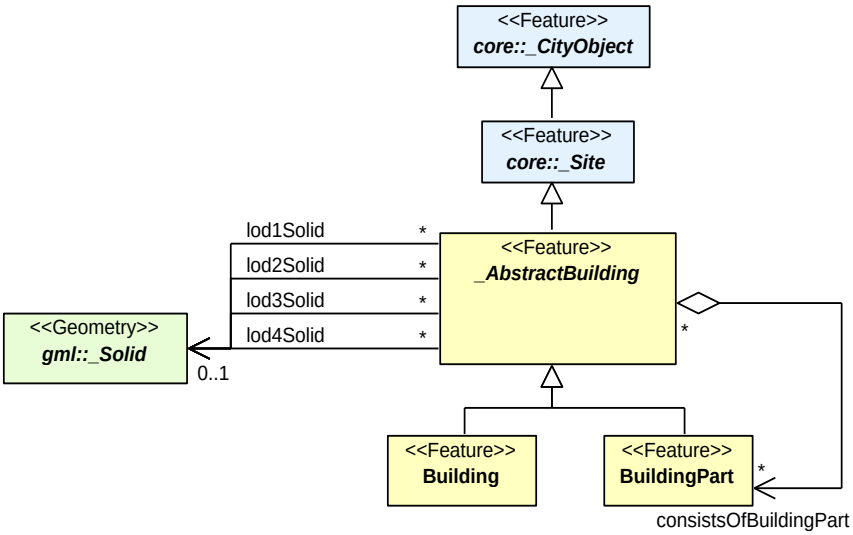

Fig. 2: Example UML diagram demonstrating the UML notation and coloring scheme used throughout the CityGML specification.

4.3 XML namespaces and namespace prefixes

The CityGML data model is thematically decomposed into a core module and thematic extension modules. All modules including the core are specified by their own XML schema file, each defining a globally unique XML namespace. The extension modules are based on the core module and, thus, contain (by reference) the CityGML core schema.

Within this document the module namespaces are associated with recommended prefixes. These prefixes are consistently used within the normative parts of this specification, for all UML diagrams and example CityGML instance documents. The CityGML core and extension modules along with their XML namespace identifiers and recommended namespace prefixes are listed in Tab. 1.

CityGML module	**Namespace identifier**	**Namespace prefix**
CityGML Core	http://www.opengis.net/citygml/2.0	core
Appearance	http://www.opengis.net/citygml/appearance/2.0	app
Bridge	http://www.opengis.net/citygml/bridge/2.0	brid
Building	http://www.opengis.net/citygml/building/2.0	bldg
CityFurniture	http://www.opengis.net/citygml/cityfurniture/2.0	frn
CityObjectGroup	http://www.opengis.net/citygml/cityobjectgroup/2.0	grp
Generics	http://www.opengis.net/citygml/generics/2.0	gen
LandUse	http://www.opengis.net/citygml/landuse/2.0	luse
Relief	http://www.opengis.net/citygml/relief/2.0	dem
Transportation	http://www.opengis.net/citygml/transportation/2.0	tran
Tunnel	http://www.opengis.net/citygml/tunnel/2.0	tun
Vegetation	http://www.opengis.net/citygml/vegetation/2.0	veg
WaterBody	http://www.opengis.net/citygml/waterbody/2.0	wtr
TexturedSurface [deprecated]	http://www.opengis.net/citygml/texturedsurface/2.0	tex

Tab. 1: List of CityGML modules, their associated XML namespace identifiers, and example namespace prefixes.

Further XML Schema definitions relevant to this standard are shown in Tab. 2 along with the corresponding XML namespace identifiers and namespace prefixes consistently used within this document.

XML Schema definition	Namespace identifier	Namespace prefix
Geography Markup Language version 3.1.1 (from OGC)	http://www.opengis.net/gml	gml
Extensible Address Language version 2.0 (from OASIS)	urn:oasis:names:tc:ciq:xsdschema:xAL:2.0	xAL
Schematron Assertion Language version 1.5	http://www.ascc.net/xml/schematron	sch

Tab. 2: List of XML Schema definitions, their associated XML namespace identifiers, and example namespace prefixes used within this document.

4.4 XML-Schema

The normative parts of the standard use the W3C XML schema language to describe the grammar of conformant CityGML data instances. XML schema is a rich language with many capabilities. While a reader who is unfamiliar with an XML schema may be able to follow the description in a general fashion, this standard is not intended to serve as an introduction to XML schema. In order to have a full understanding of this candidate standard, it is necessary for the reader to have a reasonable knowledge of XML schema.

5 Overview of CityGML

CityGML is an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is an application schema for the Geography Markup Language version 3.1.1 (GML3), the extendible international standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211.

The aim of the development of CityGML is to reach a common definition of the basic entities, attributes, and relations of a 3D city model. This is especially important with respect to the cost-effective sustainable maintenance of 3D city models, allowing the reuse of the same data in different application fields.

CityGML not only represents the graphical appearance of city models but specifically addresses the representation of the semantic and thematic properties, taxonomies and aggregations. CityGML includes a geometry model and a thematic model. The geometry model allows for the consistent and homogeneous definition of geometrical and topological properties of spatial objects within 3D city models (chapter 8). The base class of all objects is *_CityObject* which is a subclass of the GML class *_Feature*. All objects inherit the properties from *_CityObject*.

The thematic model of CityGML employs the geometry model for different thematic fields like Digital Terrain Models, sites (i.e. buildings, bridges, and tunnels), vegetation (solitary objects and also areal and volumetric biotopes), land use, water bodies, transportation facilities, and city furniture (chapter 10). Further objects, which are not explicitly modelled yet, can be represented using the concept of generic objects and attributes (chapter 6.11). In addition, extensions to the CityGML data model applying to specific application fields can be realised using the Application Domain Extensions (ADE) (chapter 6.12). Spatial objects of equal shape which appear many times at different positions like e.g. trees, can also be modelled as prototypes and used multiple times in the city model (chapter 8.2). A grouping concept allows the combination of single 3D objects, e.g. buildings to a building complex (chapter 6.8). Objects which are not geometrically modelled by closed solids can be virtually sealed in order to compute their volume (e.g. pedestrian underpasses, tunnels, or airplane hangars). They can be closed using *ClosureSurfaces* (chapter 6.4). The concept of the *TerrainIntersectionCurve* is introduced to integrate 3D objects with the Digital Terrain Model at their correct positions in order to prevent e.g. buildings from floating over or sinking into the terrain (chapter 6.5).

CityGML differentiates five consecutive Levels of Detail (LOD), where objects become more detailed with increasing LOD regarding both their geometry and thematic differentiation (chapter 6.2). CityGML files can - but do not have to - contain multiple representations (and geometries) for each object in different LOD simultaneously. Generalisation relations allow the explicit representation of aggregated objects over different scales.

In addition to spatial properties, CityGML features can be assigned appearances. Appearances are not limited to visual data but represent arbitrary observable properties of the feature's surface such as infrared radiation, noise pollution, or earthquake-induced structural stress (chapter 9).

Furthermore, objects can have external references to corresponding objects in external datasets (chapter 6.7). The possible attribute values of enumerative object attributes can be enumerated in code lists defined in external, redefinable dictionaries (chapter 6.6).

6 General characteristics of CityGML

6.1 Modularisation

The CityGML data model consists of class definitions for the most important types of objects within virtual 3D city models. These classes have been identified to be either required or important in many different application areas. However, implementations are not required to support the overall CityGML data model in order to be conformant to the standard, but may employ a subset of constructs according to their specific information needs. For this purpose, modularisation is applied to the CityGML data model (cf. chapter 7).

The CityGML data model is thematically decomposed into a *core module* and thematic *extension modules*. The core module comprises the basic concepts and components of the CityGML data model and, thus, must be implemented by any conformant system. Based on the core module, each extension covers a specific thematic field of virtual 3D city models. CityGML introduces the following thirteen thematic extension modules: *Appearance*, *Bridge*, *Building*, *CityFurniture*, *CityObjectGroup*, *Generics*, *LandUse*, *Relief*, *Transportation*, *Tunnel*, *Vegetation*, *WaterBody*, and *TexturedSurface [deprecated]*.

CityGML compliant implementations may support any combination of extension modules in conjunction with the core module. Such combinations of modules are called CityGML profiles. Therefore, CityGML profiles allow for valid partial implementations of the overall CityGML data model.

6.2 Multi-scale modelling (5 levels of detail, LOD)

CityGML supports different Levels of Detail (LOD). LODs are required to reflect independent data collection processes with differing application requirements. Further, LODs facilitate efficient visualisation and data analysis (see Fig. 3). In a CityGML dataset, the same object may be represented in different LOD simultaneously, enabling the analysis and visualisation of the same object with regard to different degrees of resolution. Furthermore, two CityGML data sets containing the same object in different LOD may be combined and integrated. However, it will be within the responsibility of the user or application to make sure objects in different LODs refer to the same real-world object.

The coarsest level LOD0 is essentially a two and a half dimensional Digital Terrain Model over which an aerial image or a map may be draped. Buildings may be represented in LOD0 by footprint or roof edge polygons. LOD1 is the well-known blocks model comprising prismatic buildings with flat roof structures. In contrast, a building in LOD2 has differentiated roof structures and thematically differentiated boundary surfaces. LOD3 denotes architectural models with detailed wall and roof structures potentially including doors and windows. LOD4 completes a LOD3 model by adding interior structures for buildings. For example, buildings in LOD4 are composed of rooms, interior doors, stairs, and furniture. In all LODs appearance information such as high-resolution textures can be mapped onto the structures (cf. 6.9).

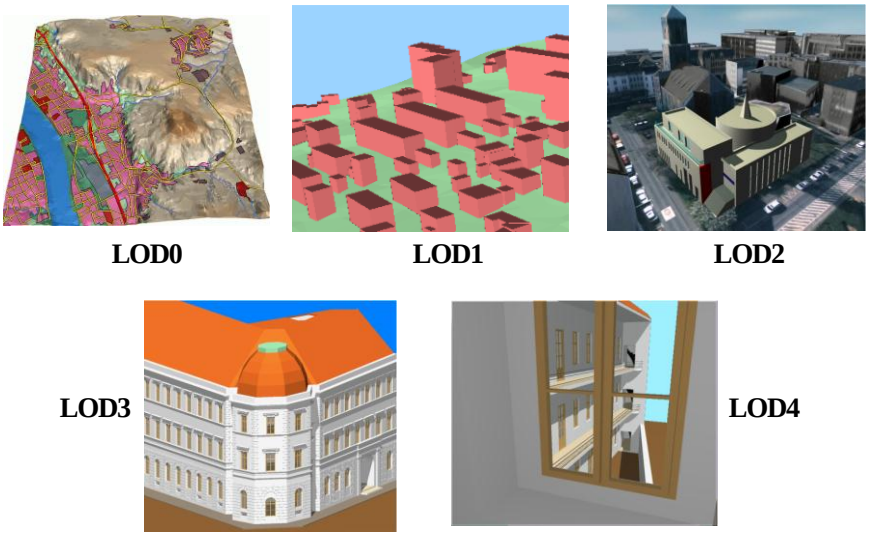

Fig. 3: The five levels of detail (LOD) defined by CityGML (source: IGG Uni Bonn)

LODs are also characterised by differing accuracies and minimal dimensions of objects (cf. Tab. 3). The accuracy requirements given in this standard are debatable and are to be considered as discussion proposals. Accuracy is described as standard deviation σ of the absolute 3D point coordinates. Relative 3D point accuracy will be added in a future version of CityGML and it is typically much higher than the absolute accuracy. In LOD1, the positional and height accuracy of points should be 5m or less, while all objects with a footprint of at least 6m by 6m should be considered. The positional and height accuracy of LOD2 is proposed to be 2m or better. In this LOD, all objects with a footprint of at least 4m × 4m should be considered. Both types of accuracies in LOD3 should be 0.5m, and the minimal footprint is suggested to be 2m × 2m. Finally, the positional and height accuracy of LOD4 should be 0.2m or less. By means of these figures, the classification in five LOD may be used to assess the quality of 3D city model datasets. The LOD categorisation makes datasets comparable and provides support for their integration.

	LOD0	**LOD1**	**LOD2**	**LOD3**	**LOD4**
Model scale description	regional, landscape	city, region	city, city districts, projects	city districts, architectural models (exterior), landmark	architectural models (interior), landmark
Class of accuracy	lowest	low	middle	high	very high
Absolute 3D point accuracy (position / height)	lower than LOD1	5/5m	2/2m	0.5/0.5m	0.2/0.2m
Generalisation	maximal generalisation	object blocks as generalised features; > 6*6m/3m	objects as generalised features; > 4*4m/2m	object as real features; > 2*2m/1m	constructive elements and openings are represented
Building installations	no	no	yes	representative exterior features	real object form
Roof structure/representation	yes	flat	differentiated roof structures	real object form	real object form
Roof overhanging parts	yes	no	yes, if known	yes	yes
CityFurniture	no	important objects	prototypes, generalized objects	real object form	real object form
SolitaryVegetationObject	no	important objects	prototypes, higher 6m	prototypes, higher 2m	prototypes, real object form
PlantCover	no	>50*50m	>5*5m	< LOD2	<LOD2
…to be continued for the other feature themes					

Tab. 3: LOD 0-4 of CityGML with their proposed accuracy requirements (discussion proposal, based on: Albert et al. 2003).

Whereas in CityGML each object can have a different representation for every LOD, often different objects from the same LOD will be generalised to be represented by an aggregate object in a lower LOD. CityGML supports the aggregation / decomposition by providing an explicit generalisation association between city objects (further details see UML diagram in chapter 10.1).

6.3 Coherent semantical-geometrical modelling

One of the most important design principles for CityGML is the coherent modelling of semantics and geometrical/topological properties. At the semantic level, real-world entities are represented by features, such as buildings, walls, windows, or rooms. The description also includes attributes, relations and aggregation hierarchies (part-whole-relations) between features. Thus the part-of-relationship between features can be derived at the semantic level only, without considering geometry. However, at the spatial level, geometry objects are assigned to features representing their spatial location and extent. So the model consists of two hierarchies: the semantic and the geometrical in which the corresponding objects are linked by relationships (cf. Stadler & Kolbe 2007). The advantage of this approach is that it can be navigated in both hierarchies and between both hierarchies arbitrarily, for answering thematic and/or geometrical queries or performing analyses.

If both hierarchies exist for a specific object, they must be coherent (i.e. it must be ensured that they match and fit together). For example, if a wall of a building has two windows and a door on the semantic level, then the geometry representing the wall must contain also the geometry parts of both windows and the door.

6.4 Closure surfaces

Objects, which are not modelled by a volumetric geometry, must be virtually closed in order to compute their volume (e.g. pedestrian underpasses or airplane hangars). They can be sealed using a *ClosureSurface*. These are

special surfaces, which are taken into account, when needed to compute volumes and are neglected, when they are irrelevant or not appropriate, for example in visualisations.

The concept of *ClosureSurface* is also employed to model the entrances of subsurface objects. Those objects like tunnels or pedestrian underpasses have to be modelled as closed solids in order to compute their volume, for example in flood simulations. The entrances to subsurface objects also have to be sealed to avoid holes in the digital terrain model (see Fig. 4). However, in close-range visualisations the entrance must be treated as open. Thus, closure surfaces are an adequate way to model those entrances.

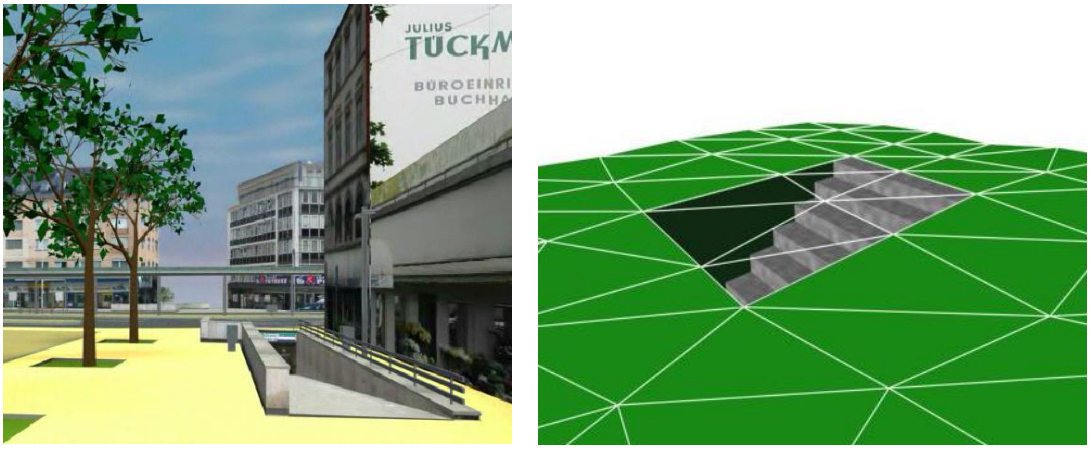

Fig. 4: Closure surfaces to seal open structures. Passages are subsurface objects (left). The entrance is sealed by a virtual *ClosureSurface*, which is both part of the DTM and the subsurface object (right) (graphic: IGG Uni Bonn).

6.5 Terrain Intersection Curve (TIC)

A crucial issue in city modelling is the integration of 3D objects and the terrain. Problems arise if 3D objects float over or sink into the terrain. This is particularly the case if terrains and 3D objects in different LOD are combined, or if they come from different providers (Kolbe and Gröger 2003). To overcome this problem, the *TerrainIntersectionCurve (TIC)* of a 3D object is introduced. These curves denote the exact position, where the terrain touches the 3D object (see Fig. 5). TICs can be applied to buildings and building parts (cf. chapter 10.3), bridge, bridge parts and bridge construction elements (cf. chapter 10.5), tunnel and tunnel parts (cf. chapter 10.4), city furniture objects (cf. chapter 10.9), and generic city objects (cf. chapter 10.12). If, for example, a building has a courtyard, the TIC consists of two closed rings: one ring representing the courtyard boundary, and one which describes the building's outer boundary. This information can be used to integrate the building and a terrain by 'pulling up' or 'pulling down' the surrounding terrain to fit the *TerrainIntersectionCurve*. The DTM may be locally warped to fit the TIC. By this means, the TIC also ensures the correct positioning of textures or the matching of object textures with the DTM. Since the intersection with the terrain may differ depending on the LOD, a 3D object may have different *TerrainIntersectionCurves* for all LOD.

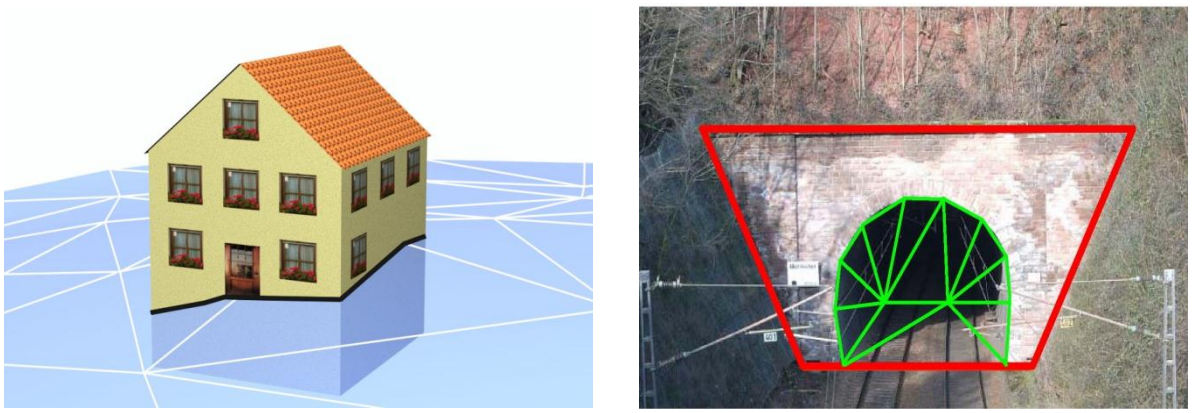

Fig. 5: *TerrainIntersectionCurve* for a building (left, black) and a tunnel object (right, white). The tunnel's hollow space is sealed by a triangulated *ClosureSurface* (graphic: IGG Uni Bonn).

6.6 Code lists for enumerative attributes

CityGML feature types often include attributes whose values can be enumerated in a list of discrete values. An example is the attribute *roof type* of a building, whose attribute values typically are saddle back roof, hip roof, semi-hip roof, flat roof, pent roof, or tent roof. If such an attribute is typed as string, misspellings or different names for the same notion obstruct interoperability. Moreover, the list of possible attribute values often is not fixed and may substantially vary for different countries (e.g., due to national law and regulations) and for different information communities.

In CityGML, such enumerative attributes are of type *gml:CodeType* and their allowed attribute values can be provided in a *code list* which is specified outside the CityGML schema. A code list contains coded attribute values and ensures that the same code is used for the same notion or concept. If a code list is provided for an enumerative attribute, the attribute may only take values from this list. This allows applications to validate the attribute value and thus facilitates semantic and syntactic interoperability. It is recommended that code lists are implemented as *simple dictionaries* following the *GML 3.1.1 Simple Dictionary Profile* (cf. Whiteside 2005).

The governance of code lists is decoupled from the governance of the CityGML schema and specification. Thus, code lists may be specified by any organisation or information community according to their information needs. There shall be one authority per code list who is in charge of the code list values and the maintenance of the code list. Further information on the CityGML code list mechanism is provided in chapter 10.14.

Code lists can have references to existing models. For example, room codes defined by the Open Standards Consortium for Real Estate (OSCRE) can be referenced or classifications of buildings and building parts introduced by the National Building Information Model Standard (NBIMS) can be used. Annex C contains non-normative code lists proposed by the SIG 3D for almost all enumerative attributes in CityGML. They can be directly referenced in CityGML instance documents and serve as an example for the definition of code lists.

6.7 External references

3D objects are often derived from or have relations to objects in other databases or data sets. For example, a 3D building model may have been constructed from a two-dimensional footprint in a cadastre data set, or may be derived from an architectural model (Fig. 6). The reference of a 3D object to its corresponding object in an external data set is essential, if an update must be propagated or if additional data is required, for example the name and address of a building's owner in a cadastral information system or information on antennas and doors in a facility management system. In order to supply such information, each *_CityObject* may refer to external data sets (for the UML diagram see Fig. 21; and for XML schema definition see annex A.1) using the concept of *ExternalReference*. Such a reference denotes the external information system and the unique identifier of the object in this system. Both are specified as a Uniform Resource Identifier (URI), which is a generic format for references to any kind of resources on the internet. The generic concept of external references allows for any *_CityObject* an arbitrary number of links to corresponding objects in external information systems (e.g. ALKIS, ATKIS, OS MasterMap®, GDF, etc.).

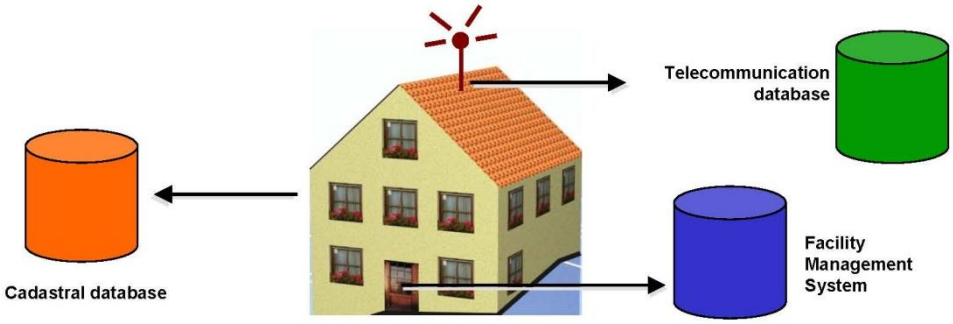

Fig. 6: External references (graphic: IGG Uni Bonn).

6.8 City object groups

The grouping concept of CityGML allows for the aggregation of arbitrary city objects according to user-defined criteria, and to represent and transfer these aggregations as part of a city model (for the UML diagram see

chapter 10.11; XML schema definition see annex A.6). A group may be assigned one or more names and may be further classified by specific attributes, for example, "*escape route from room no. 43 in house no. 1212 in a fire scenario*" as a name and "*escape route*" as type. Each member of the group can optionally be assigned a role name, which specifies the role this particular member plays in the group. This role name may, for example, describe the sequence number of this object in an escape route, or in the case of a building complex, denote the main building.

A group may contain other groups as members, allowing nested grouping of arbitrary depth. The grouping concept is delivered by the thematic extension module *CityObjectGroup* of CityGML (cf. chapter 10.11).

6.9 Appearances

Information about a surface's appearance, i.e. observable properties of the surface, is considered an integral part of virtual 3D city models in addition to semantics and geometry. Appearance relates to any surface-based theme, e.g. infrared radiation or noise pollution, not just visual properties. Consequently, data provided by appearances can be used as input for both presentation of and analysis in virtual 3D city models.

CityGML supports feature appearances for an arbitrary number of themes per city model. Each LOD of a feature can have an individual appearance. Appearances can represent – among others – textures and georeferenced textures. CityGML's appearance model is packaged within its own extension module *Appearance* (cf. chapter 9).

6.10 Prototypic objects / scene graph concepts

In CityGML, objects of equal shape like trees and other vegetation objects, traffic lights and traffic signs can be represented as prototypes which are instantiated multiple times at different locations (Fig. 7). The geometry of prototypes is defined in local coordinate systems. Every instance is represented by a reference to the prototype, a base point in the world coordinate reference system and a transformation matrix that facilitates scaling, rotation, and translation of the prototype. The principle is adopted from the concept of scene graphs used in computer graphics standards like VRML and X3D. As the GML3 geometry model does not provide support for scene graph concepts, it is implemented as an extension to the GML3 geometry model (for further description cf. chapter 8.2).

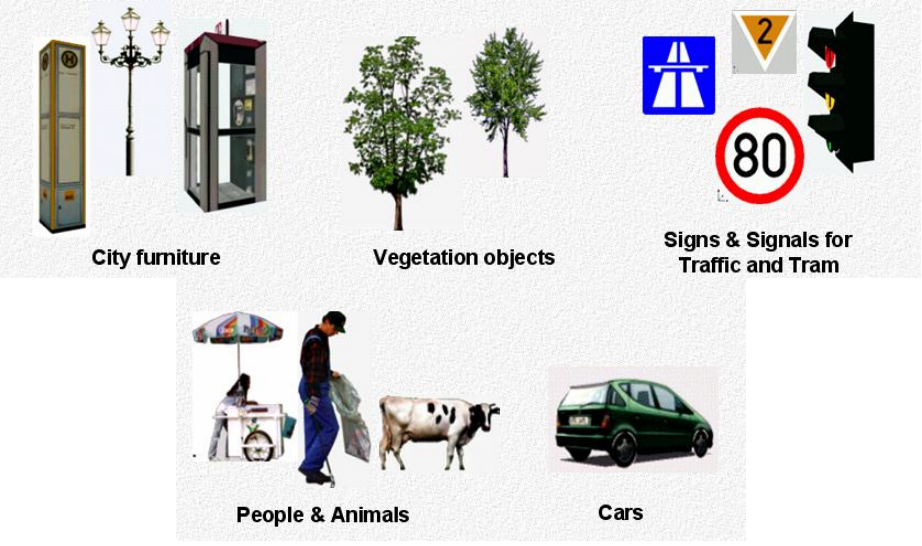

Fig. 7: Examples of prototypic shapes (source: Rheinmetall Defence Electronics).

6.11 Generic city objects and attributes

CityGML is being designed as a universal topographic information model that defines object types and attributes which are useful for a broad range of applications. In practical applications the objects within specific 3D city models will most likely contain attributes which are not explicitly modelled in CityGML. Moreover, there might be 3D objects which are not covered by the thematic classes of CityGML. CityGML provides two different

concepts to support the exchange of such data: 1) generic objects and attributes, and 2) Application Domain Extensions (cf. chapter 6.12).

The concept of generic objects and attributes allows for the extension of CityGML applications during runtime, i.e. any *_CityObject* may be augmented by additional attributes, whose names, data types, and values can be provided by a running application without any change of the CityGML XML schema. Similarly, features not represented by the predefined thematic classes of the CityGML data model may be modelled and exchanged using generic objects. The generic extensions of CityGML are provided by the thematic extension module *Generics* (cf. chapter 10.12).

The current version of CityGML does not include, for example, explicit thematic models for embankments, excavations and city walls. These objects may be stored or exchanged using generic objects and attributes.

6.12 Application Domain Extensions (ADE)

Application Domain Extensions (ADE) specify additions to the CityGML data model. Such additions comprise the introduction of new properties to existing CityGML classes like e.g. the number of habitants of a building or the definition of new object types. The difference between ADEs and generic objects and attributes is, that an ADE has to be defined in an extra XML schema definition file with its own namespace. This file has to explicitly import the XML Schema definition of the extended CityGML modules.

The advantage of this approach is that the extension is formally specified. Extended CityGML instance documents can be validated against the CityGML and the respective ADE schema. ADEs can be defined (and even standardised) by information communities which are interested in specific application fields. More than one ADE can be actively used in the same dataset (further description cf. chapter 10.13).

ADEs may be defined for one or even several CityGML modules providing a high flexibility in adding additional information to the CityGML data model. Thus, the ADE mechanism is orthogonally aligned with the modularisation approach of CityGML. Consequently, there is no separate extension module for ADEs.

In this specification, two examples for ADEs are included:

- An ADE for Noise Immission Simulation (Annex H) which is employed in the simulation of environmental noise dispersion according to the Environmental Noise Directive of the European Commission (2002/49/EC);
- An ADE for Ubiquitous Network Robots Services (Annex I) which demonstrates the usage of CityGML for the navigation of robots in indoor environments.

Further examples for ADEs are the *CAFM ADE* (Bleifuß et al., 2009) for facility management, the *UtilityNetworkADE* (Becker et al., 2011) for the integrated 3D modeling of multi-utility networks and their interdependencies, the *HydroADE* (Schulte and Coors, 2008) for hydrographical applications and the *GeoBIM (IFC) ADE* (van Berlo et al., 2011) which combines BIM information from IFC (from bSI) with CityGML and is implemented in the open source modelserver BIMserver.org.

7 Modularisation

CityGML is a rich standard both on the thematic and geometric-topological level of its data model. On its thematic level CityGML defines classes and relations for the most relevant topographic objects in cities and regional models comprising built structures, elevation, vegetation, water bodies, city furniture, and more. In addition to geometry and appearance content these thematic components allow to employ virtual 3D city models for sophisticated analysis tasks in different application domains like simulations, urban data mining, facility management, and thematic inquiries.

CityGML is to be seen as a framework giving geospatial 3D data enough space to grow in geometrical, topological and semantic aspects over its lifetime. Thus, geometry and semantics of city objects may be flexibly structured covering purely geometric datasets up to complex geometric-topologically sound and spatio-semantically coherent data. By this means, CityGML defines a single object model and data exchange format applicable to consecutive process steps of 3D city modelling from geometry acquisition, data qualification and refinement to preparation of data for specific end-user applications, allowing for iterative data enrichment and lossless information exchange (cf. Kolbe et al. 2009).

According to this idea of a framework, applications are not required to support all thematic fields of CityGML in order to be compliant to the standard, but may employ a subset of constructs corresponding to specific relevant requirements of an application domain or process step. The use of logical subsets of CityGML limits the complexity of the overall data model and explicitly allows for valid partial implementations. As for version 2.0 of the CityGML standard, possible subsets of the data model are defined and embraced by so called CityGML modules. A CityGML module is an aggregate of normative aspects that must all be implemented as a whole by a conformant system. CityGML consists of a core module and thematic extension modules.

The CityGML core module defines the basic concepts and components of the CityGML data model. It is to be seen as the universal lower bound of the overall CityGML data model and a dependency of all thematic extension modules. Thus, the core module is unique and must be implemented by any conformant system. Based on the CityGML core module, each extension module contains a logically separate thematic component of the CityGML data model. The extensions to the core are derived by vertically slicing the overall CityGML data model. Since the core module is contained (by reference) in each extension module, its general concepts and components are universal to all extension modules. The following thirteen thematic extension modules are introduced by version 2.0 of the CityGML standard. They are directly related to clauses of this document each covering the corresponding thematic field of CityGML:

- Appearance (cf. clause 9),
- Bridge (cf. clause 10.5)
- Building (cf. clause 10.3),
- CityFurniture (cf. clause 10.9),
- CityObjectGroup (cf. clause 10.11),
- Generics (cf. clause 10.12),
- LandUse (cf. clause 10.10),
- Relief (cf. clause 10.2),
- Transportation (cf. clause 10.7),
- Tunnel (cf. clause 10.4)
- Vegetation (cf. clause 10.8),
- WaterBody (cf. clause 10.6), and
- TexturedSurface [deprecated] (cf. clause 9.8).

The thematic decomposition of the CityGML data model allows for implementations to support any combination of extension modules in conjunction with the core module in order to be CityGML conformant. Thus, the extension modules may be arbitrarily combined according to the information needs of an application or application domain. A combination of modules is called a CityGML profile. The union of all modules is defined as the

CityGML *base profile*. The base profile is unique at any given time and forms the upper bound of the overall CityGML data model. Any other CityGML profile must be a valid subset of the base profile. By following the concept of CityGML modules and profiles, valid partial implementations of the CityGML data model may be realised in a well-defined way.

As for future development, each CityGML module may be further developed independently from other modules by expert groups and information communities. Resulting proposals and changes to modules may be introduced into future revisions of the CityGML standard without affecting the validity of other modules. Furthermore, thematic components not covered by the current CityGML data model may be added to future revisions of the standard by additional thematic extension modules. These additional extensions may establish dependency relations to any other existing CityGML module but shall at least be dependent on the CityGML core module. Consequently, the CityGML base profile may vary over time as new extensions are added. However, if a specific application has information needs to be modelled and exchanged which are beyond the scope of the CityGML data model, this application data can also be incorporated within the existing modules using CityGML's *Application Domain Extension* mechanism (cf. clause 10.13) or by employing the concepts of generic city objects and attributes (cf. chapter 10.12).

The introduced modularisation approach supports CityGML's versatility as a data modelling framework and exchange format addressing various application domains and different steps of 3D city modelling. For sake of clarity, applications should announce the level of conformance to the CityGML standard by declaring the employed CityGML profile. Since the core module is part of all profiles, this should be realised by enumerating the implemented thematic extension modules. For example, if an implementation supports the *Building* module, the *Relief* module, and the *Vegetation* module in addition to the core, this should be announced by "CityGML [Building, Relief, Vegetation]". In case the base profile is supported, this should be indicated by "CityGML [full]".

7.1 CityGML core and extension modules

Each CityGML module is specified by its own XML Schema definition file and is defined within an individual and globally unique XML target namespace. According to dependency relations between modules, each module may, in addition, import namespaces associated to such related CityGML modules. However, a single namespace shall not be directly included in two modules. Thus, all elements belonging to one module are associated to the module's namespace only. By this means, module elements are guaranteed to be properly separated and distinguishable in CityGML instance documents.

Compared to CityGML versions before 1.0, the aforementioned namespace conventions introduce an extra level of complexity to data files as there is no single CityGML namespace any more. In contrast, components of different CityGML modules and, thus, of different namespaces may be arbitrarily mixed within the same CityGML instance document. Furthermore, an application might have to parse instance documents containing elements of modules which are not employed by the application itself. These parsing problems though can easily be overcome by non-"schema-aware" applications, i.e. applications that do not parse and interpret GML application schemas in a generic way. Elements from different namespaces than those declared by the application's employed CityGML profile could be skipped. Comparable observations have to be made when using CityGML's Application Domain Extension mechanism (cf. clause 10.13).

As for version 2.0 of the CityGML standard, there are no two thematic extension modules related by dependency. Thus, all extension modules are truly independent from each other and may be separately supported by implementations. However, the CityGML core module is a dependency for any extension module. This means that the XML schema file of the core module is imported by each XML schema file defining an extension.

The dependency relations between CityGML's modules are illustrated in Fig. 8 using an UML package diagram. Each module is represented by a package. The package names correspond to the module names. A dashed arrow in the figure indicates that the schema at the tail of the arrow depends upon the schema at the head of the arrow. For CityGML modules, a dependency occurs where one schema <import>s another schema and accordingly the corresponding XML namespace. For example, the extension module *Building* imports the schema of the *CityGML Core* module. A short description of each module is given in Tab. 4.

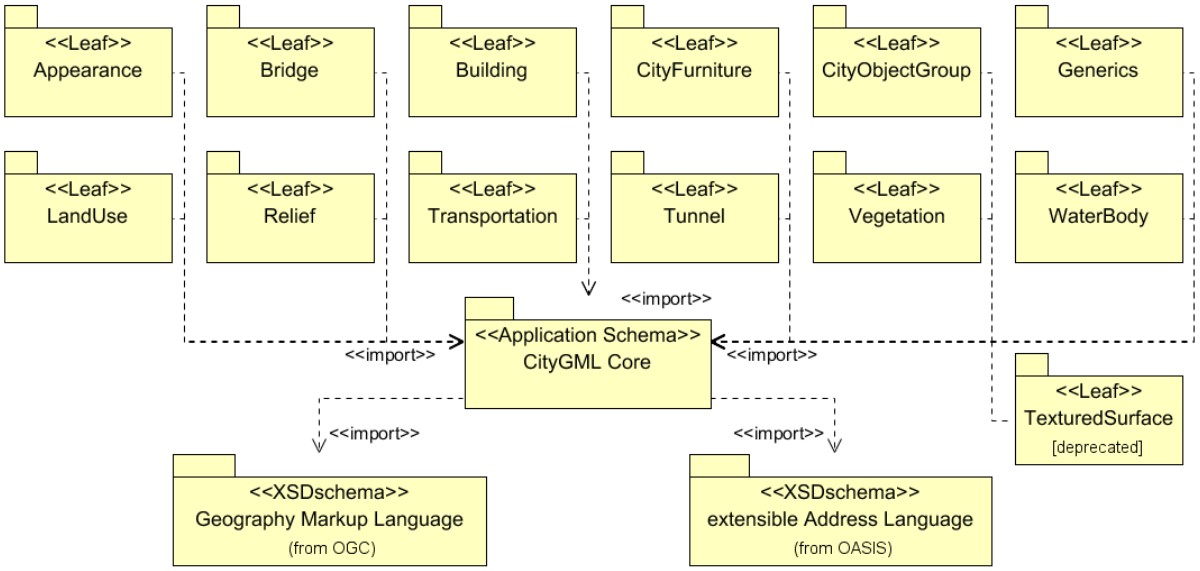

Fig. 8: UML package diagram illustrating the separate modules of CityGML and their schema dependencies. Each extension module (indicated by the leaf packages) further imports the GML 3.1.1 schema definition in order to represent spatial properties of its thematic classes. For readability reasons, the corresponding dependencies have been omitted.

Module name	CityGML Core
XML namespace identifier	http://www.opengis.net/citygml/2.0
XML Schema file	cityGMLBase.xsd
Recommended namespace prefix	core
Module description	The *CityGML Core* module defines the basic components of the CityGML data model. Primarily, this comprises abstract base classes from which all thematic classes are (transitively) derived. But also non-abstract content common to more than one extension module, for example basic data types, is defined within the core module. The core module itself imports the XML schema definition files of GML version 3.1.1 and the OASIS extensible Address Language xAL.

Module name	Appearance
XML namespace identifier	http://www.opengis.net/citygml/appearance/2.0
XML Schema file	appearance.xsd
Recommended namespace prefix	app
Module description	The *Appearance* module provides the means to model appearances of CityGML features, i.e. observable properties of the feature's surface. Appearance data may be stored for each city object. Therefore, the abstract base class *_CityObject* defined within the core module is augmented by an additional property using CityGML's *Application Domain Extension* mechanism. Thus, the *Appearance* module has a deliberate impact on all thematic extension modules.

Modul name	Bridge
XML namespace identifier	http://www.opengis.net/citygml/bridge/2.0
XML Schema file	bridge.xsd
Recommended namespace prefix	brid
Module description	The *Bridge* module allows the representation of thematic and spatial aspects of bridges, bridge parts, bridge installations, and interior bridge structures in four levels of detail (LOD 1 – 4).

Module name	Building
XML namespace identifier	http://www.opengis.net/citygml/building/2.0
XML Schema file	building.xsd
Recommended namespace prefix	bldg
Module description	The *Building* module allows the representation of thematic and spatial aspects of buildings, building parts, building installations, and interior building structures in five levels of detail (LOD 0 – 4).

Module name	CityFurniture
XML namespace identifier	http://www.opengis.net/citygml/cityfurniture/2.0
XML Schema file	cityFurniture.xsd
Recommended namespace prefix	frn
Module description	The *CityFurniture* module is used to represent city furniture objects in cities. City furniture objects are immovable objects like lanterns, traffic signs, advertising columns, benches, or bus stops that can be found in traffic areas, residential areas, on squares, or in built-up areas.

Module name	CityObjectGroup
XML namespace identifier	http://www.opengis.net/citygml/cityobjectgroup/2.0
XML Schema file	cityObjectGroup.xsd
Recommended namespace prefix	grp
Module description	The *CityObjectGroup* module provides a grouping concept for CityGML. Arbitrary city objects may be aggregated in groups according to user-defined criteria to represent and transfer these aggregations as part of the city model. A group may be further classified by specific attributes.

Module name	Generics
XML namespace identifier	http://www.opengis.net/citygml/generics/2.0
XML Schema file	generics.xsd
Recommended namespace prefix	gen
Module description	The *Generics* module provides generic extensions to the CityGML data model that may be used to model and exchange additional attributes and features not covered by the predefined thematic classes of CityGML. However, generic extensions shall only be used if appropriate thematic classes or attributes are not provided by any other CityGML module. In order to represent generic attributes, the *Generics* module augments the abstract base class *_CityObject* defined within the core module by an additional property using CityGML's *Application Domain Extension* mechanism. Thus, the *Generics* module has a deliberate impact on all thematic extension modules.

Module name	LandUse
XML namespace identifier	http://www.opengis.net/citygml/landuse/2.0
XML Schema file	landUse.xsd
Recommended namespace prefix	luse
Module description	The *LandUse* module allows for the representation of areas of the earth's surface dedicated to a specific land use.

Module name	Relief
XML namespace identifier	http://www.opengis.net/citygml/relief/2.0
XML Schema file	relief.xsd
Recommended namespace prefix	dem
Module description	The *Relief* module allows for the representation of the terrain in a city model. CityGML supports terrain representations in different levels of detail, reflecting different accuracies or resolutions. The terrain may be specified as a regular raster or grid, as a TIN, by break lines, and by mass points.

Module name	Transportation
XML namespace identifier	http://www.opengis.net/citygml/transportation/2.0
XML Schema file	transportation.xsd
Recommended namespace prefix	tran
Module description	The *Transportation* module is used to represent the transportation features within a city, for example roads, tracks, railways, or squares. Transportation features may be represented as a linear network or by geometrically describing their 3D surfaces.

Module Name	Tunnel
XML namespace identifier	http://www.opengis.net/citygml/tunnel/2.0
XML Schema file	tunnel.xsd
Recommended namespace prefix	tun
Module description	The *Tunnel* module facilitates the representation of thematic and spatial aspects of tunnels, tunnel parts, tunnel installations, and interior tunnel structures in four level of detail (LOD 1 – 4)

Module name	Vegetation
XML namespace identifier	http://www.opengis.net/citygml/vegetation/2.0
XML Schema file	vegetation.xsd
Recommended namespace prefix	veg
Module description	The *Vegetation* module provides thematic classes to represent vegetation objects. CityGML's vegetation model distinguishes between solitary vegetation objects like trees, and vegetation areas which represent biotopes like forests or other plant communities.

Module name	WaterBody
XML namespace identifier	http://www.opengis.net/citygml/waterbody/2.0
XML Schema file	waterBody.xsd
Recommended namespace prefix	wtr
Module description	The *WaterBody* module represents the thematic aspects and 3D geometry of rivers, canals, lakes, and basins. It does, however, not inherit any hydrological or other dynamic aspects so far.

Module name	TexturedSurface [deprecated]
XML namespace identifier	http://www.opengis.net/citygml/texturedsurface/2.0
XML Schema file	texturedSurface.xsd
Recommended namespace prefix	tex
Module description	The *TexturedSurface* module allows for assigning visual appearance properties (color, shininess, transparency) and textures to 3D surfaces. Due to inherent limitations of its modelling approach this module has been marked deprecated and is expected to be removed in future CityGML versions. Appearance information provided by this module can be converted to CityGML's *Appearance* module without information loss. Thus, the use of the *TexturedSurface* module is *strongly discouraged*.

Tab. 4: Overview of CityGML's core and thematic extensions modules.

7.2 CityGML profiles

A CityGML profile is a combination of thematic extension modules in conjunction with the core module of CityGML. Each CityGML instance document shall employ the CityGML profile appropriate to the provided data. In general, two approaches to employ a CityGML profile within an instance document can be differentiated:

1. **CityGML profile definition embedded inline the CityGML instance document**
   A CityGML profile can be bound to an instance document using the *schemaLocation* attribute defined in the XML Schema instance namespace, http://www.w3.org/2001/XMLSchema-instance (commonly associated with the prefix *xsi*). The *xsi:schemaLocation* attribute provides a way to locate the XML Schema definition for namespaces defined in an XML instance document. Its value is a whitespace-delimited list of pairs of Uniform Resource Identifiers (URIs) where each pair consists of a namespace followed by the location of that namespace's XML Schema definition, which is typically a .xsd file.

   By this means, the namespaces of the respective CityGML modules shall be defined within a CityGML instance document. The *xsi:schemaLocation* attribute then shall be used to provide the location to the respective XML Schema definition of each module. All example instance documents given in Annex G follow this first approach.

2. **CityGML profile definition provided by a separate XML Schema definition file**
   The CityGML profile may also be specified by its own XML Schema file. This schema file shall combine the appropriate CityGML modules by importing the corresponding XML Schema definitions. For this purpose, the *import* element defined in the XML Schema namespace shall be used, http://www.w3.org/2001/XMLSchema (commonly associated with the prefix *xs*). For the *xs:import* element, the namespace of the imported CityGML module along with the location of the namespace's XML Schema definition have to be declared. In order to apply a CityGML profile to an instance document, the profile's schema has to be bound to the instance document using the *xsi:schemaLocation* attribute. The XML Schema file of the CityGML profile shall not contain any further content.

   The *targetNamespace* of the profile's schema shall differ from the namespaces of the imported CityGML modules. The namespace associated with the profile should be in control of the originator of the instance document and must be given as a previously unused and globally unique URI. The profile's XML Schema file must be available (or accessible on the internet) to everybody parsing the associated CityGML instance document.

The second approach is illustrated by the following example XML Schema definition for the base profile of CityGML. Since the base profile is the union of all CityGML modules, the corresponding XML Schema definition imports each and every CityGML module. By this means, all components of the CityGML data model are available in and may be exchanged by instance documents referencing this example base profile. The schema definition file of the base profile is shipped with the CityGML schema package, and is accessible at http://schemas.opengis.net/citygml/profiles/base/2.0/CityGML.xsd.

```
<xs:schema xmlns="http://www.opengis.net/citygml/profiles/base/2.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.opengis.net/citygml/profiles/base/2.0" elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="2.0.0">
  <xs:import namespace="http://www.opengis.net/citygml/appearance/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/appearance/2.0/appearance.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/bridge/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/bridge/2.0/bridge.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/building/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/building/2.0/building.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/cityfurniture/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/cityfurniture/2.0/cityFurniture.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/cityobjectgroup/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/cityobjectgroup/2.0/cityObjectGroup.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/generics/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/generics/2.0/generics.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/landuse/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/landuse/2.0/landUse.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/relief/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/relief/2.0/relief.xsd"/>
```



```
  <xs:import namespace="http://www.opengis.net/citygml/transportation/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/transportation/2.0/transportation.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/tunnel/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/tunnel/2.0/tunnel.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/vegetation/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/vegetation/2.0/vegetation.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/waterbody/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/waterbody/2.0/waterBody.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/texturedsurface/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/texturedsurface/2.0/texturedSurface.xsd"/>
</xs:schema>
```

The following excerpt of a CityGML dataset exemplifies how to apply the base profile schema *CityGML.xsd* to a CityGML instance document. The dataset contains two building objects and a city object group. The base profile defined by *CityGML.xsd* is referenced using the *xsi:schemaLocation* attribute of the root element. Thus, all CityGML modules are employed by the instance document and no further references to the XML Schema documents of the CityGML modules are necessary.

```
<core:CityModel xmlns="http://www.opengis.net/citygml/profiles/base/2.0"
  xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xmlns:grp="http://www.opengis.net/citygml/cityobjectgroup/2.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/citygml/profiles/base/2.0
                      http://schemas.opengis.net/citygml/profiles/base/2.0/CityGML.xsd">
  <core:cityObjectMember>
    <bldg:Building gml:id="Build0815">
      <core:externalReference>
        <core:informationSystem>http://www.adv-online.de</core:informationSystem>
        <core:externalObject>
          <core:uri>urn:adv:oid:DEHE123400007001</core:uri>
        </core:externalObject>
      </core:externalReference>
      <bldg:function
          codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_function.xml">1000</bldg:function>
      <bldg:yearOfConstruction>1985</bldg:yearOfConstruction>
      <bldg:roofType
          codeSpace="http://www.sig3d.org/codelists/standard//building/2.0/_AbstractBuilding_roofType.xml">1030</bldg:roofType>
      <bldg:measuredHeight uom="#m">8.0</bldg:measuredHeight>
      <bldg:storeysAboveGround>2</bldg:storeysAboveGround>
      <bldg:storeyHeightsAboveGround uom="#m">2.5 2.5</bldg:storeyHeightsAboveGround>
      <bldg:lod2Solid> ... </bldg:lod2Solid>
    </bldg:Building>
  </core:cityObjectMember>
  <core:cityObjectMember>
    <bldg:Building gml:id="Build0817">
      …
    </bldg:Building>
  </core:cityObjectMember>
  <core:cityObjectMember>
    <grp:CityObjectGroup gml:id="Complex113">
      <gml:name>Hotel complex 'Scenic View'</gml:name>
      <grp:function>building group</grp:function>
      <grp:groupMember role="main building" xlink:href="#Build0817"/>
      <grp:groupMember xlink:href="#Build0815"/>
    </grp:CityObjectGroup>
  </core:cityObjectMember>
</core:CityModel>
```

8 Spatial model

Spatial properties of CityGML features are represented by objects of GML3's geometry model. This model is based on the standard ISO 19107 'Spatial Schema' (Herring 2001), representing 3D geometry according to the well-known *Boundary Representation* (B-Rep, cf. Foley et al. 1995). CityGML actually uses only a subset of the GML3 geometry package, defining a profile of GML3. This subset is depicted in Fig. 9 and Fig. 10. Furthermore, GML3's explicit Boundary Representation is extended by scene graph concepts, which allow the representation of the geometry of features with the same shape implicitly and thus more space efficiently (chapter 8.2).

8.1 Geometric-topological model

The geometry model of GML3 consists of primitives, which may be combined to form complexes, composite geometries or aggregates. For each dimension, there is a geometrical primitive: a zero-dimensional object is a *Point*, a one-dimensional a *_Curve*, a two-dimensional a *_Surface*, and a three-dimensional a *_Solid* (Fig. 9). Each geometry can have its own coordinate reference system. A solid is bounded by surfaces and a surface by curves. In CityGML, a curve is restricted to be a straight line, thus only the GML3 class *LineString* is used. Surfaces in CityGML are represented by *Polygons*, which define a planar geometry, i.e. the boundary and all interior points are required to be located in one single plane.

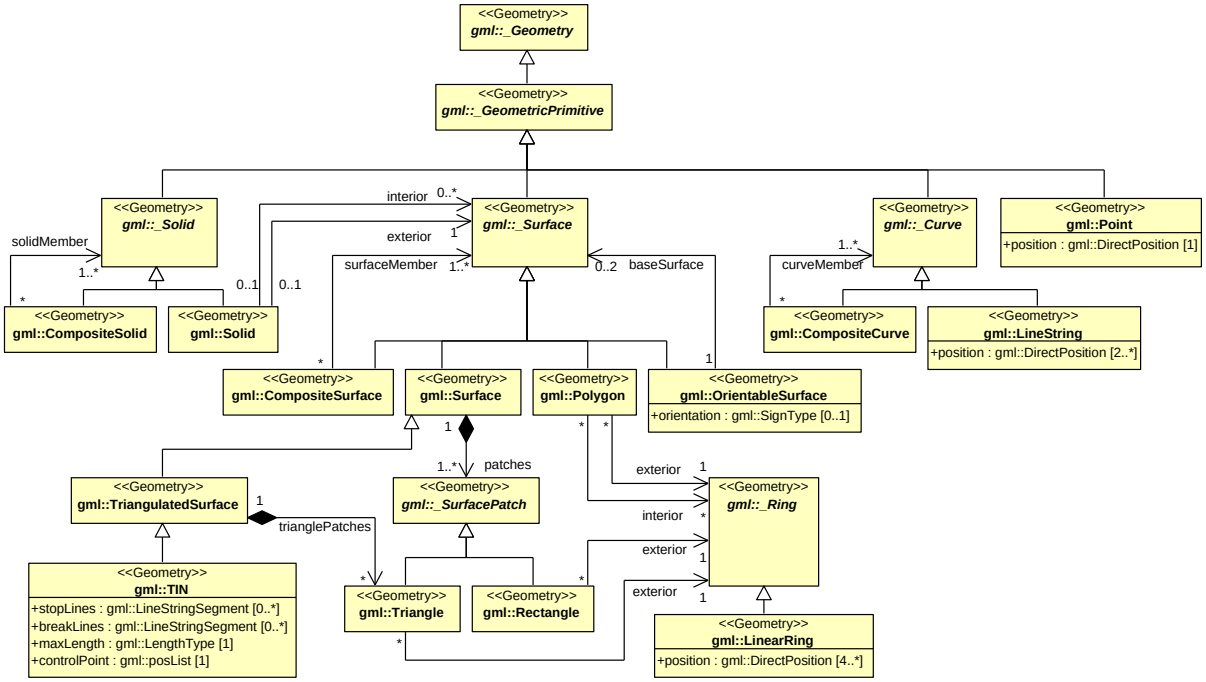

Fig. 9: UML diagram of CityGML's geometry model (subset and profile of GML3): Primitives and Composites.

Combined geometries can be aggregates, complexes or composites of primitives (see illustration in Fig. 11). For an aggregate, the spatial relationship between components is not restricted. They may be disjoint, overlapping, touching, or disconnected. GML3 provides a special aggregate for each dimension, a *MultiPoint*, a *MultiCurve*, *a MultiSurface*, and a *MultiSolid* (see Fig. 10). In contrast to aggregates, a complex is topologically structured: its parts must be disjoint, must not overlap and are allowed to touch, at most, at their boundaries or share parts of their boundaries. A composite is a special complex provided by GML3. It can only contain elements of the same dimension. Its elements must be disjoint as well, but they must be topologically connected along their boundaries. A composite can be a *CompositeSolid*, a *CompositeSurface, or CompositeCurve.* (cf. Fig. 9).

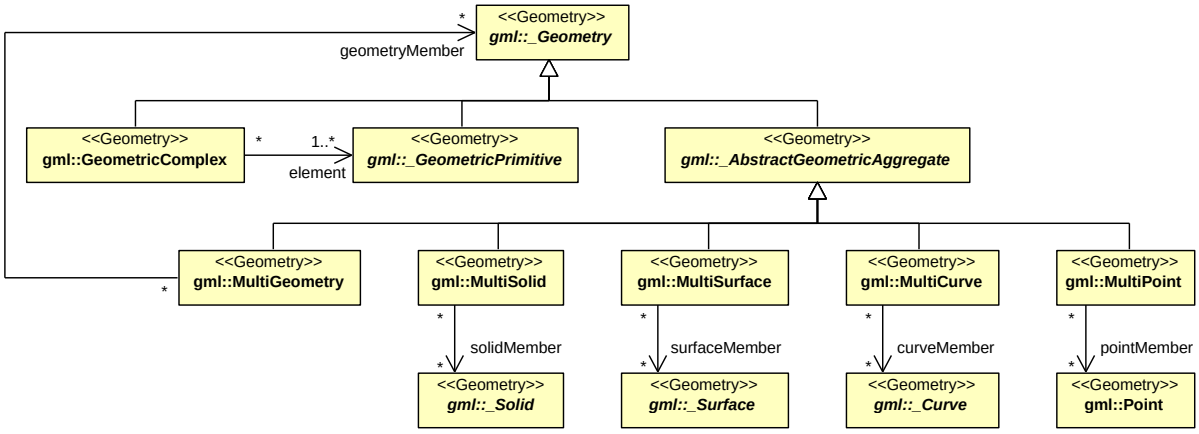

Fig. 10: UML diagram of CityGML's geometry model: Complexes and Aggregates

An *OrientableSurface* is a surface with an explicit orientation, i.e. two sides, front and back, can be distinguished. This may be used to assign textures to specific sides of a surface, or to distinguish the exterior and the interior side of a surface when bounding a solid. Please note, that curves and surfaces have a default orientation in GML which results from the order of the defining points. Thus, an *OrientableSurface* only has to be used, if the orientation of a given GML geometry has to be reversed.

TriangulatedSurfaces are special surfaces, which specify triangulated irregular networks often used to represent the terrain. While a *TriangulatedSurface* is a composition of explicit *Triangles*, the subclass *TIN* is used to represent a triangulation in an implicit way by a set of control points, defining the nodes of the triangles. The triangulation may be reconstructed using standard triangulation methods (Delaunay triangulation). In addition, break lines and stop lines define contour characteristics of the terrain.

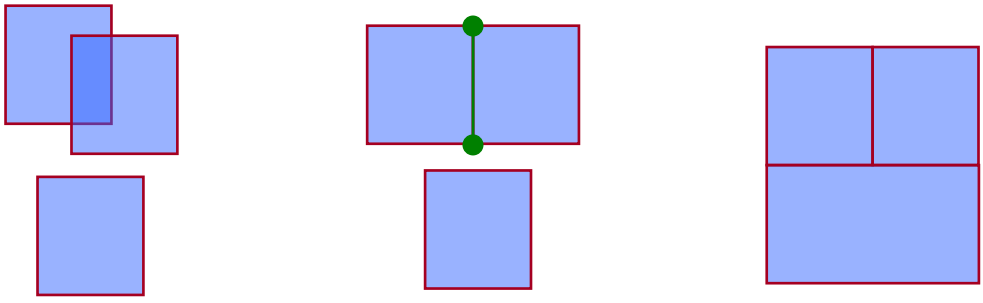

Fig. 11: Combined geometries.

The GML3 composite model realises a recursive aggregation schema for every primitive type of the corresponding dimension. This aggregation schema allows the definition of nested aggregations (hierarchy of components). For example, a building geometry (*CompositeSolid*) can be composed of the house geometry (*CompositeSolid*) and the garage geometry (*Solid*), while the house's geometry is further decomposed into the roof geometry (*Solid*) and the geometry of the house body (*Solid*).

CityGML provides the explicit modelling of topology, for example the sharing of geometry objects between features or other geometries. One part of space is represented only once by a geometry object and is referenced by all features or more complex geometries which are defined or bounded by this geometry object. Thus redundancy is avoided and explicit topological relations between parts are maintained. Basically, there are three cases. First, two features may be defined spatially by the same geometry. For example, if a path is both a transportation feature and a vegetation feature, the surface geometry defining the path is referenced both by the transportation object and by the vegetation object. Second, geometry may be shared between a feature and another geometry. A geometry defining a wall of a building may be referenced twice: by the solid geometry defining the geometry of the building, and by the wall feature. Third, two geometries may reference the same geometry, which is in the boundary of both. For example, a building and an adjacent garage may be represented by two solids. The surface describing the area where both solids touch may be represented only once and it is referenced by both solids. As

it can be seen from Fig. 12, this requires partitioning of the respective surfaces. In general, Boundary Representation only considers visible surfaces. However, to make topological adjacency explicit and to allow the possibility of deletion of one part of a composed object without leaving holes in the remaining aggregate touching elements are included. Whereas touching is allowed, permeation of objects is not in order to avoid the multiple representation of the same space. However, the use of topology in CityGML is optional.

In order to implement topology, CityGML uses the XML concept of *XLinks* provided by GML. Each geometry object that should be shared by different geometric aggregates or different thematic features is assigned an unique identifier, which may be referenced by a GML geometry property using a *href* attribute. CityGML does not deploy the built-in topology package of GML3, which provides separate topology objects accompanying the geometry. This kind of topology is very complex and elaborate. Nevertheless, it lacks flexibility when data sets, which might include or neglect topology, should be covered by the same data model. The XLink topology is simple and flexible and nearly as powerful as the explicit GML3 topology model. However, a disadvantage of the XLink topology is that navigation between topologically connected objects can only be performed in one direction (from an aggregate to its components), not (immediately) bidirectional as it is the case for GML's built-in topology. An example for CityGML's topology representation is given in the dataset listed in annex G.4.

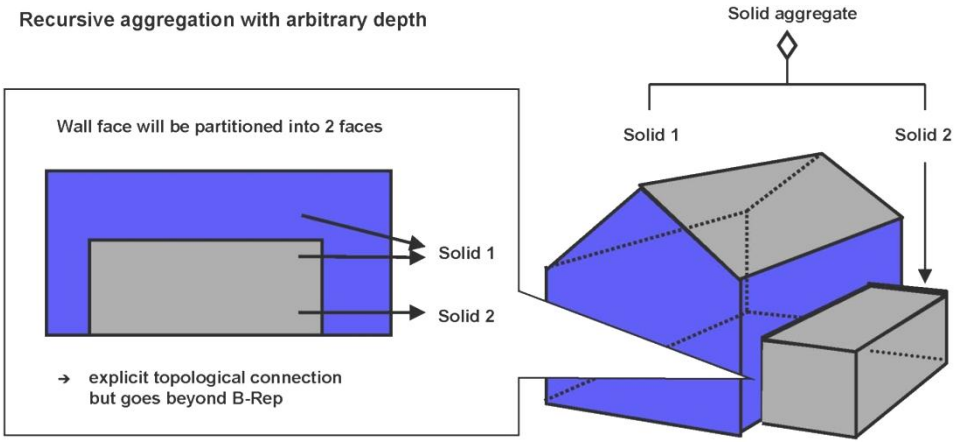

Fig. 12: Recursive aggregation of objects and geometries in CityGML (graphic: IGG Uni Bonn).

The following excerpt of a CityGML example file defines a *gml:Polygon* with a gml:id *wallSurface4711*, which is part of the geometry property *lod2Solid* of a building. Another building being adjacent to the first building references this shared polygon in its geometry representation.

```
<bldg:Building>
  <bldg:lod2Solid>
    <gml:surfaceMember>
      <gml:Polygon gml:id="wallSurface4711">
        <gml:exterior>
          <gml:LinearRing>
            <gml:pos srsDimension="3">32.0 31.0 2.5</gml:pos>
            ...
          </gml:LinearRing>
        </gml:exterior>
      </gml:Polygon>
    </gml:surfaceMember>
    ...
  </bldg:lod2Solid>
</bldg:Building>
...
<bldg:Building>
  <bldg:lod2Solid>
    <gml:surfaceMember>
      <gml:OrientableSurface orientation="-">
        <gml:baseSurface xlink:href="#wallSurface4711"/>
      </gml:OrientableSurface>
    </gml:surfaceMember>
    ...
  </bldg:lod2Solid>
</bldg:Building>
```


8.2 Spatial reference system

When dealing with geoinformation and virtual 3D city models in particular, the exact spatial reference is of utmost importance and a key requirement for the integration of different spatial datasets in a single 3D city model. CityGML inherits GML3's spatial capabilities of handling Coordinate Reference Systems (CRS) which is the usual way of denoting the spatial reference in GML 3.1.1. As CityGML is a true 3D standard, geometry elements are associated with a 3D CRS. There are only few exceptions to this rule where CityGML allows a 2D geometry element (for example, the *referencePoint* of a *GeoreferencedTexture* defined in CityGML's *Appearance* module must be given with 2D coordinate values, cf. chapter 9.4).

In general, a geometry may point to the CRS definition used by this geometry through the attribute *srsName* which is inherited from the abstract GML superclass *gml:_Geometry*. This may be a reference to a well-known CRS definition provided by an authority organization such as the European Petroleum Survey Group (EPSG), but may also be a pointer to a CRS that is locally defined within the same CityGML instance document. The OGC document "Definition identifier URNs in OGC namespace" (cf. Whiteside 2009; OGC Doc. No. 07-092r3) provides best practices for the URN encoding of CRS references. Amongst others, it describes how to reference a single well-known 3D CRS definition (such as a 3D geographic CRS) as well as a compound CRS which combines two or more well-known CRS definitions (e.g., a projected CRS for the planimetry with a vertical CRS for the height reference). Examples for denoting a compound CRS for a CityGML instance document are given in Annex G.

GML3 also supports the definition of engineering CRSs which are used in a contextually local sense. For example, this might be a local 3D Cartesian coordinate system that is essentially based on a flat-earth approximation of the earth's surface, and thus ignores the effect of earth curvature on feature geometry (cf. chapter 12.1.4.4 of the GML 3.1.1 specification document). Local engineering CRSs are commonly applied in the AEC/FM domain and thus are useful when integrating CAD data or BIM models into a 3D city model. Annex G.9 provides an example demonstrating the definition of an engineering CRS within a CityGML instance document and the use of local coordinate values for the feature geometry. The definition of an engineering CRS requires an anchor point which relates the origin of the local coordinate system to a point on the earth's surface in order to facilitate the transformation of coordinates from the local engineering CRS.

According to GML 3.1.1, if no *srsName* attribute is given on a geometry element, then the CRS shall be specified as part of the larger context this geometry element is part of, e.g. a geometric aggregate. For convenience in constructing feature and feature collection instances, the value of the *srsName* attribute on the *gml:Envelope* (or *gml:Box*) which is the value of the *gml:boundedBy* property of the feature shall be inherited by all directly expressed geometries in all properties of the feature or members of the collection, unless overruled by the presence of a local *srsName.* Thus it is not necessary for a geometry to carry an *srsName* attribute if it uses the same CRS as given on the *gml:boundedBy* property of its parent feature. Inheritance of the CRS continues to any depth of nesting, but if overruled by a local *srsName* declaration, then the new CRS is inherited by all its children in turn (cf. chapter 8.3 of the GML 3.1.1 specification document).

It is *strongly recommended* that any CityGML instance document explicitly specifies the CRS for all contained geometry elements. This is especially important if the instance document is to be exchanged externally with third parties or is to be integrated with other spatial datasets. A mixed usage of different CRSs within the same dataset is possible and conformant with GML 3.1.1, whereas a single CRS reference given on the embracing *CityModel* feature collection (cf. chapter 10.1) simplifies the processing of the dataset by software systems. As for CityGML 2.0, this recommendation is non-normative and thus not accompanied by a conformance class. The main reason for this is to maintain backwards compatibility with CityGML 1.0.

8.3 Implicit geometries, prototypic objects, scene graph concepts

The concept of implicit geometries is an enhancement of the geometry model of GML3. It is, for example, used in CityGML's building, bridge, tunnel, and vegetation model as well as for city furniture and generic objects. Implicit geometries may be applied to features from different thematic fields of CityGML in order to geometrically represent the features within a specific level of detail (LOD). Thus, each extension module may define spatial properties providing implicit geometries for its thematic classes. For this reason, the concept of implicit geometries is defined within the CityGML core module (cf. chapter 10.1). However, its description is drawn here

since implicit geometries are part of CityGML's spatial model. The UML diagram is depicted in Fig. 13. The corresponding XML schema definition is provided in annex A.1.

An implicit geometry is a geometric object, where the shape is stored only once as a prototypical geometry, for example a tree or other vegetation objects, a traffic light or a traffic sign. This prototypic geometry object is re-used or referenced many times, wherever the corresponding feature occurs in the 3D city model. Each occurrence is represented by a link to the prototypic shape geometry (in a local cartesian coordinate system), by a transformation matrix that is multiplied with each 3D coordinate of the prototype, and by an anchor point denoting the base point of the object in the world coordinate reference system. This reference point also defines the CRS to which the world coordinates belong after the application of the transformation. In order to determine the absolute coordinates of an implicit geometry, the anchor point coordinates have to be added to the matrix multiplication results. The transformation matrix accounts for the intended rotation, scaling, and local translation of the prototype. It is a 4x4 matrix that is multiplied with the prototype coordinates using homogeneous coordinates, i.e. (x,y,z,1). This way even a projection might be modelled by the transformation matrix.

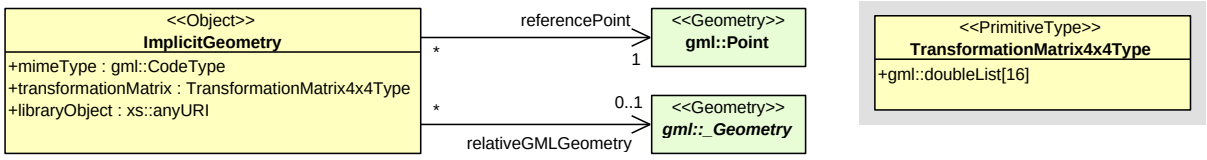

Fig. 13: UML diagram of *ImplicitGeometries*. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the *CityGML Core* module.

The reason for using the concept of implicit geometries in CityGML is space efficiency. Since the shape of, for example, trees of the same species can be treated as identical, it would be inefficient to model the detailed geometry of each of the large number of trees explicitly. The concept of implicit geometries is similar to the well known concept of *primitive instancing* used for the representation of *scene graphs* in the field of computer graphics (Foley et al. 1995).

The term *implicit geometry* refers to the principle that a geometry object with a complex shape can be simply represented by a base point and a transformation, implicitly unfolding the object's shape at a specific location in the world coordinate system.

The shape of an *ImplicitGeometry* can be represented in an external file with a proprietary format, e.g. a VRML file, a DXF file, or a 3D Studio MAX file. The reference to the implicit geometry can be specified by an URI pointing to a local or remote file, or even to an appropriate web service. Alternatively, the shape can be defined by a GML3 geometry object. This has the advantage that it can be stored or exchanged inline within the CityGML dataset. Typically, the shape of the geometry is defined in a local coordinate system where the origin lies within or near to the object's extent. If the shape is referenced by an URI, also the MIME type of the denoted object has to be specified (e.g. "model/vrml" for VRML models or "model/x3d+xml" for X3D models).

The implicit representation of 3D object geometry has some advantages compared to the explicit modelling, which represents the objects using absolute world coordinates. It is more space-efficient, and thus more extensive scenes can be stored or handled by a system. The visualisation is accelerated since 3D graphics cards support the scene graph concept. Furthermore, the usage of different shape versions of objects is facilitated, e.g. different seasons, since only the library objects have to be exchanged (see example in Fig. 65).

XML namespace

The XML namespace of the *CityGML Core* module defining the concept of implicit geometries is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/2.0. Within the XML Schema definition of the core module, this URI is also used to identify the default namespace.

ImplicitGeometryType, ImplicitRepresentationPropertyType

```
<xs:complexType name="ImplicitGeometryType">
  <xs:complexContent>
    <xs:extension base="gml:AbstractGMLType">
      <xs:sequence>
        <xs:element name="mimeType" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="transformationMatrix" type="TransformationMatrix4x4Type" minOccurs="0"/>
```



```
        <xs:element name="libraryObject" type="xs:anyURI" minOccurs="0"/>
        <xs:element name="relativeGMLGeometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="referencePoint" type="gml:PointPropertyType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="ImplicitGeometry" type="ImplicitGeometryType" substitutionGroup="gml:_GML"/>
<!-- ========================================================================================== -->
<xs:complexType name="ImplicitRepresentationPropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="ImplicitGeometry"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
```

8.3.1 Code lists

The *mimeType* attribute of *ImplicitGeometry* is specified as *gml:CodeType*. The values of this property can be enumerated in a code list. A proposal for this code list can be found in annex C.6.

8.3.2 Example CityGML datasets

An example for an implicit geometry is given by the following city furniture object (cf. chapter 10.9), which is represented by a geometry in LOD2:

```
<frn:CityFurniture>
  <!-- class “traffic”; as specified in the code list proposed by the SIG 3D (cf. annex C.4) -->
  <frn:class codeSpace="http://www.sig3d.org/codelists/standard/cityfurniture/2.0/CityFurniture_class.xml">1000</frn:class>
  <!-- function “traffic light”; as specified in the code list proposed by the SIG 3D (cf. annex C.4) -->
  <frn:function codeSpace="http://www.sig3d.org/codelists/standard/cityfurniture/2.0/CityFurniture_function.xml">1080</frn:function>
  <frn:lod2ImplicitRepresentation>
    <core:ImplicitGeometry>
      <core:mimeType>model/vrml</core:mimeType>
      <core:libraryObject>
        http://www.some-3d-library.com/3D/furnitures/TrafficLight434.wrl
      </core:libraryObject>
      <core:referencePoint>
        <gml:Point srsName="urn:ogc:def:crs,crs:EPSG:6.12:31467,crs:EPSG:6.12:5783">
          <gml:pos srsDimension="3">5793898.77 3603845.54 44.8</gml:pos>
        </gml:Point>
      </core:referencePoint>
    </core:ImplicitGeometry>
  </frn:lod2ImplicitRepresentation>
</frn:CityFurniture>
```

The shape of the geometry of the traffic light (city furniture with class “1000” and function “1080” according to the code lists proposed in annex C.4) is defined by a VRML file which is specified by a URL. This library object, which is defined in a local coordinate system, is transformed to its actual location by adding the coordinates of the reference point.

The following clip of a CityGML file provides a more complex example for an implicit geometry:

```
<frn:CityFurniture>
  <!-- class “traffic”; as specified in the code list proposed by the SIG 3D (cf. annex C.4) -->
  <frn:class>1000</frn:class>
  <!-- function “traffic light”; as specified in the code list proposed by the SIG 3D (cf. annex C.4) -->
  <frn:function>1080</frn:function>
  <frn:lod2ImplicitRepresentation>
    <core:ImplicitGeometry>
      <core:mimeType>model/vrml</core:mimeType>
      <core:transformationMatrix>
        0.866025   -0.5   0   0.7
        0.5   0.866025   0   0.8
        0   0   1   0
        0   0   0   1
      </core:transformationMatrix>
      <core:libraryObject>
```



```
        http://www.some-3d-library.com/3D/furnitures/TrafficLight434.wrl
      </core:libraryObject>
      <core:referencePoint>
        <gml:Point srsName="urn:ogc:def:crs,crs:EPSG:6.12:31467,crs:EPSG:6.12:5783">
          <gml:pos srsDimension="3">5793898.77 3603845.54 44.8</gml:pos>
        </gml:Point>
      </core:referencePoint>
    </core:ImplicitGeometry>
  </frn:lod2ImplicitRepresentation>
</frn:CityFurniture>
```

In addition to the first example, a transformation matrix is specified. It is a homogeneous matrix, serialized in a row major fashion, i.e. the first four entries in the list denote the first row of the matrix, etc. The matrix combines a translation by the vector (0.7, 0.8, 0) – the origin of the local reference system is not the center of the object – and a rotation around the z-axis by 30 degrees (*cos(30) = 0.866025* and *sin(30) = 0.5*). This rotation is necessary to align the traffic light with respect to a road. The actual position of the traffic light is computed as follows:

1. each point of the VRML file (with homogeneous coordinates) is multiplied by the transformation matrix;
2. for each resulting point, the reference point $(5793898.77, 3603845.54, 44.8, 1)^T$ is added, yielding the actual geometry of the city furniture.

8.3.3 Conformance requirements

Base requirements

1. In order to geometrically represent a feature using the concept of implicit geometries, the corresponding thematic class of the feature shall define a spatial property of the type *ImplicitRepresentationPropertyType*. Thus, for all CityGML extension modules only the type *ImplicitRepresentationPropertyType* shall be used for spatial properties providing implicit geometries.
2. If the shape of an implicit geometry is referenced by an URI using the *libraryObject* property (type: *xs:anyURI*) of the element *ImplicitGeometry*, also the MIME type of the denoted object must be specified.

Referential integrity

3. The type *ImplicitRepresentationPropertyType* may contain an *ImplicitGeometry* element inline or an XLink reference to a remote *ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the corresponding property of type *ImplicitRepresentationPropertyType* may only point to a remote *ImplicitGeometry* element (where remote *ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

9 Appearance model

In addition to spatial properties, CityGML features have *appearances* – observable properties of the feature's surface. Appearances are not limited to visual data but represent arbitrary categories called *themes* such as infrared radiation, noise pollution, or earthquake-induced structural stress. Each LOD can have an individual appearance for a specific theme. An appearance is composed of data for each surface geometry object, i.e. *surface data*. A single surface geometry object may have surface data for multiple themes. Similarly, surface data can be shared by multiple surface geometry objects (e.g. road paving). Finally, surface data values can either be constant across a surface or depend on the exact location within the surface.

CityGML's appearance model is defined within the extension module *Appearance* (cf. chapter 7). The UML diagram of the appearance model is illustrated in Fig. 14, for the XML Schema definition see annex A.2.

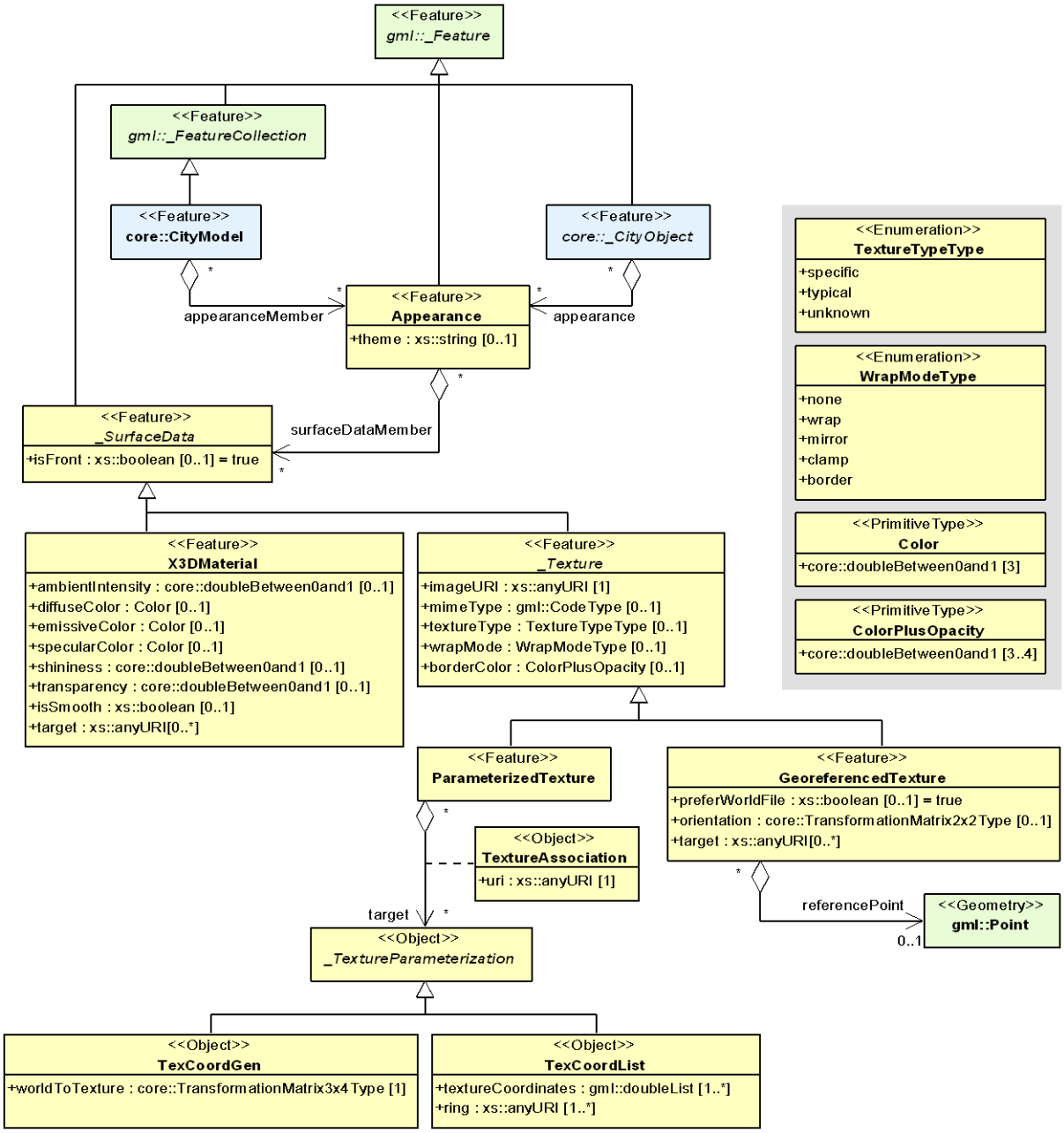

Fig. 14: UML diagram of CityGML's appearance model. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Appearance* module.

In CityGML's appearance model, themes are represented by an identifier only. The appearance of a city model for a given theme is defined by a set of *Appearance* objects referencing this theme. Thus, the *Appearance* objects

belonging to the same theme compose a virtual group. They may be included in different places within a CityGML dataset. Furthermore a single CityGML dataset may contain several themes. An *Appearance* object collects surface data relevant for a specific theme either for individual features or the whole city model in any LOD. Surface data is represented by objects of class *_SurfaceData* and its descendents with each covering the whole area of a surface geometry object. The relation between surface data and surface geometry objects is expressed by an *URI* (*Uniform Resource Identifier*) link from a *_SurfaceData* object to an object of type *gml:AbstractSurfaceType* or type *gml:MultiSurface*.

A constant surface property is modelled as material. A surface property, which depends on the location within the surface, is modelled as texture. Each surface geometry object can have both a material and a texture per theme and side. This allows for providing both a constant approximation and a complex measurement of a surface's property simultaneously. An application is responsible for choosing the appropriate property representation for its task (e.g. analysis or rendering). A specific mixing is not defined since this is beyond the scope of CityGML. If a surface geometry object is to receive multiple textures or materials, each texture or material requires a separate theme. The mixing of themes or their usage is not defined within CityGML and left to the application.

XML namespace

The XML namespace of the CityGML *Appearance* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/appearance/2.0. Within the XML Schema definition of the *Appearance* module, this URI is also used to identify the default namespace.

9.1 Relation between appearances, features and geometry

Despite the close relation between surface data and surface, surface data is stored separately in the feature to preserve the original GML geometry model. Instead of surface data being an attribute of the respective target surface geometry object, each surface data object maintains a set of URIs specifying the *gml:id*s of the target surface geometry objects (of type *gml:AbstractSurfaceType* or *gml:MultiSurface*). In case of a composite or aggregate target surface, the surface data object is assigned to all contained surfaces. Other target types such as features, solids, or *gml:AbstractSurfacePatchType* (which includes *gml:Triangle*) are invalid, even though the XML schema language cannot formally express constrains on URI target types. For the exact mapping function of surface data values to a surface patch refer to the respective surface data type description.

The limitation of valid target types to *gml:AbstractSurfaceType* and *gml:MultiSurface* excluding *gml:AbstractSurfacePatchType* is based on the GML geometry model and its use in CityGML. In general, GML surfaces are represented using subclasses of *gml:AbstractSurfaceType*. Such surfaces are required to be continuous. A *gml:MultiSurface* does not need to fulfill this requirement and consequently is no *gml:AbstractSurfaceType* (cf. 8.1). Since captured real-world surfaces often cannot be guaranteed to be continuous, CityGML allows for *gml:MultiSurface* to represent a feature's boundary in various places as an alternative to a continuous surface. To treat such surfaces similarly to a *gml:CompositeSurface*, surface data objects are allowed to link to *gml:MultiSurface* objects. *gml:AbstractSurfacePatchType* is no valid target type since it is not derived from *gml:AbstractGMLType*. Thus, a *gml:AbstractSurfacePatchType* (which includes *gml:Triangle* and *gml:Rectangle*) cannot receive a *gml:id* and cannot be referenced.

Each surface geometry object can have per theme at most one active front-facing material, one active back-facing material, one active front-facing texture, and one active back-facing texture. If multiple surface data objects of the same category and theme are assigned to a surface geometry object, one is chosen to become active. Multiple indirect assignments due to nested surface definitions are resolved by overwriting, e.g. the front-facing material of a *gml:Polygon* becomes active by overwriting the front-facing material of the parental *gml:CompositeSurface*. Multiple direct assignments, i.e. a surface geometry object's *gml:id* is referenced multiple times within a theme, are not allowed and are resolved implementation-dependently by choosing exactly one of the conflicting surface data objects. Thus, multiple direct assignments within a theme need to be avoided.

Each *_CityObject* feature can store surface data. Thus, surface data is arranged in the feature hierarchy of a CityGML dataset. Surface data then links to its target surface using URIs. Even though the linking mechanism permits arbitrary links across the feature hierarchy to another feature's surface, it is recommended to follow the principle of locality: Surface data should be stored such that the linked surfaces only belong to the containing *_CityObject* feature and its children. "Global" surface data should be stored with the city model. Adhering to the

locality principle also ensures that CityObjects retrieved from a WFS will contain the respective appearance information.

The locality principle allows for the following algorithm to find all relevant *_SurfaceData* objects referring to a given surface geometry object (of type *gml:AbstractSurfaceType* or *gml:MultiSurface*) in a given *_CityObject*:

```
function findSurfaceData
  in:  gmlSurface, cityObject
  out: frontMaterial, frontTexture, backMaterial, backTexture
1:  frontMaterial := empty
2:  frontTexture := empty
3:  backMaterial := empty
4:  backTexture := empty
5:  flip := false
6:
7:  while (gmlSurface) {   // traverse the geometry hierarchy from inner to outer
8:    cObj := cityObject   // start from the innermost cityobject
9:
10:   while (cObj) {   // traverse the cityobject hierarchy for the current geometry object
11:     // search all surfaceData objects in all appearance containers
12:     foreach (appearance in cObj) {
13:       foreach (surfaceData in appearance) {
14:         if (surfaceData refers to gmlSurface) {   // if a surfaceData object refers to the geometry object, check its category
15:           if (flip) {   // consider flipping
16:             // only pick the first surfaceData for a particular category
17:             if (surfaceData is frontside material AND backMaterial is empty) {
18:               backMaterial := surfaceData
19:             }
20:             if (surfaceData is frontside texture AND backTexture is empty) {
21:               backTexture := surfaceData
22:             }
23:             if (surfaceData is backside material AND frontMaterial is empty) {
24:               frontMaterial := surfaceData
25:             }
26:             if (surfaceData is backside texture AND frontTexture is empty) {
27:               frontTexture := surfaceData
28:             }
29:           } else {
30:             // only pick the first surfaceData for a particular category
31:             if (surfaceData is frontside material AND frontMaterial is empty) {
32:               frontMaterial := surfaceData
33:             }
34:             if (surfaceData is frontside texture AND frontTexture is empty) {
35:               frontTexture := surfaceData
36:             }
37:             if (surfaceData is backside material AND backMaterial is empty) {
38:               backMaterial := surfaceData
39:             }
40:             if (surfaceData is backside texture AND backTexture is empty) {
41:               backTexture := surfaceData
42:             }
43:           }
44:
45:           // shortcut: could stop here if all 4 categories have been found
46:         }
47:       }
48:     }
49:     cObj := cObj.parent   // this also includes the global CityModel
50:   }
51:   gmlSurface := gmlSurface.parent   // this also includes a root gml:MultiSurface
52:   if (gmlSurface isA gml:OrientableSurface AND gmlSurface.orientation is negative) {
53:     negate flip
54:   }
55: }
```

Listing 1: Algorithm to find all relevant *_SurfaceData* objects referring to a given surface geometry object (of type *gml:AbstractSurfaceType* or *gml:MultiSurface*) in a given *_CityObject.*

The evaluation of the *isFront* property of a *_SurfaceData* object needs to take *gml:OrientableSurface*s into account, as those can flip the orientation of a surface. Assume a *gml:OrientableSurface os*, which flips its base surface *bs*. A front side texture *t* targeting *bs* will appear on the actual front side of *bs*. If *t* targets *os*, it will appear on the back side of *bs*. If *t* targets both *os* and *bs*, it appears on both sides of *bs* since it becomes the front and back side texture.

XLinks influence the hierarchy traversal in the pseudocode. In general, the separation of surface data and geometry objects requires the reevaluation of the surface data assignment for each occurrence of a geometry object in the context of the respective *_CityObject*. Stepping up the (geometry or *_CityObject*) hierarchy in the algorithm takes XLinks into account, i.e., for the purpose of this algorithm, referenced objects are conceptually copied to the location of the referring XLink. In particular, this applies to *ImplicitGeometry* objects. If an *ImplicitGeometry* object contains GML geometry (in the *relativeGMLGeometry* property), the surface data assignment needs to be reevaluated in the context of each referring *_CityObject*. Thus, the appearance (but not the relative geometry) of a given *ImplicitGeometry* can differ between its occurrences. A consistent appearance results if all required surface data objects are placed in *Appearance* objects and the latter are stored either

1. in the *_CityObject* containing the original *ImplicitGeometry* with XLinks referencing the same *Appearance* objects in all *_CityObject*s that refer to the *ImplicitGeometry* or
2. in the global *CityModel*.

9.2 Appearance and SurfaceData

The feature class *Appearance* defines a container for surface data objects. It provides the *theme* that all contained surface data objects are related to. All appearance objects with the same theme in a CityGML file are considered a group. Surface data objects are stored in the *surfaceDataMember* property. They can be used in multiple themes simultaneously as remote properties.

The feature class *_SurfaceData* is the base class for materials and textures. Its only element is the boolean flag *isFront*, which determines the side a surface data object applies to. Please note, that all classes of the appearance model support CityGML's ADE mechanism (cf. chapters 6.12 and 10.13). The hooks for application specific extensions are realized by the elements "*_GenericApplicationPropertyOf…*".

AppearanceType, Appearance, AppearancePropertyType

```
<xs:complexType name="AppearanceType">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureType">
      <xs:sequence>
        <xs:element name="theme" type="xs:string" minOccurs="0"/>
        <xs:element name="surfaceDataMember" type="SurfaceDataPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfAppearance" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Appearance" type="AppearanceType" substitutionGroup="gml:_Feature"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfAppearance" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="AppearancePropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="Appearance"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
```

appearanceMember, appearance

```
<xs:element name="appearanceMember" type="gml:FeaturePropertyType" substitutionGroup="gml:featureMember"/>
<!-- ========================================================================== -->
<xs:element name="appearance" type="AppearancePropertyType" substitutionGroup="core:_GenericApplicationPropertyOfCityObject"/>
```

The definition of *appearanceMember* allows for an arbitrary or even mixed sequence of *_CityObject* features and *Appearance* features within a *CityModel* feature collection (cf. chapter 10.1).

In order to store appearance information within a single *_CityObject* feature, the corresponding abstract class *_CityObject* of the core module is augmented by the property element *appearance*. The additional property *appearance* is injected into *_CityObject* using CityGML's *Application Domain Extension* mechanism (cf. chapter 10.13). By this means, each thematic subclass of *_CityObject* inherits this property. Thus, the *Appearance* module has a deliberate impact on each extension module defining thematic subclasses of *_CityObject*.

AbstractSurfaceDataType, _SurfaceData, SurfaceDataPropertyType

```
<xs:complexType name="AbstractSurfaceDataType" abstract="true">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureType">
      <xs:sequence>
        <xs:element name="isFront" type="xs:boolean" default="true" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfSurfaceData" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_SurfaceData" type="AbstractSurfaceDataType" abstract="true" substitutionGroup="gml:_Feature"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfSurfaceData" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="SurfaceDataPropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="_SurfaceData" minOccurs="0"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
```

9.3 Material

Materials define light reflection properties being constant for a whole surface geometry object. The definition of the class *X3DMaterial* is adopted from the X3D and COLLADA specification (cf. X3D, COLLADA specification). *diffuseColor* defines the color of diffusely reflected light. *specularColor* defines the color of a directed reflection. *emissiveColor* is the color of light generated by the surface. All colors use RGB values with red, green, and blue between 0 and 1. Transparency is defined separately using the *transparency* element where 0 stands for fully opaque and 1 for fully transparent. *ambientIntensity* defines the minimum percentage of *diffuseColor* that is visible regardless of light sources. *shininess* controls the sharpness of the specular highlight. 0 produces a soft glow while 1 results in a sharp highlight. *isSmooth* gives a hint for normal interpolation. If this boolean flag is set to true, vertex normals should be used for shading (Gouraud shading). Otherwise, normals should be constant for a surface patch (flat shading).

Target surfaces are specified using *target* elements. Each element contains the URI of one target surface geometry object (of type *gml:AbstractSurfaceType* or *gml:MultiSurface*).

X3DMaterialType, X3DMaterial

```
<xs:complexType name="X3DMaterialType">
  <xs:complexContent>
    <xs:extension base="AbstractSurfaceDataType">
      <xs:sequence>
        <xs:element name="ambientIntensity" type="core:doubleBetween0and1" default="0.2" minOccurs="0"/>
        <xs:element name="diffuseColor" type="Color" default="0.8 0.8 0.8" minOccurs="0"/>
        <xs:element name="emissiveColor" type="Color" default="0.0 0.0 0.0" minOccurs="0"/>
        <xs:element name="specularColor" type="Color" default="1.0 1.0 1.0" minOccurs="0"/>
        <xs:element name="shininess" type="core:doubleBetween0and1" default="0.2" minOccurs="0"/>
        <xs:element name="transparency" type="core:doubleBetween0and1" default="0.0" minOccurs="0"/>
        <xs:element name="isSmooth" type="xs:boolean" default="false" minOccurs="0"/>
        <xs:element name="target" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfX3DMaterial" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
```



```
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="X3DMaterial" type="X3DMaterialType" substitutionGroup="_SurfaceData"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfX3DMaterial" type="xs:anyType" abstract="true"/>
```

9.4 Texture and texture mapping

The abstract base class for textures is *_Texture*. Textures in CityGML are always raster-based 2D textures. The raster image is specified by *imageURI* using a URI and can be an arbitrary image data resource, even a preformatted request for a web service. The image data format can be defined using standard MIME types in the *mimeType* element.

Textures can be qualified by the attribute *textureType*. The *textureType* differentiates between textures, which are specific for a certain object (*specific*) and prototypic textures being typical for that object surface (*typical)*. Textures may also be classified as *unknown*.

The specification of texture wrapping is adopted from the COLLADA standard. Texture wrapping is required when accessing a texture outside the underlying image raster. *wrapMode* can have one of five values (Fig. 15 illustrates the effect of these wrap modes):

1. *none* – the resulting color is fully transparent
2. *wrap* – the texture is repeated
3. *mirror* – the texture is repeated and mirrored
4. *clamp* – the texture is clamped to its edges
5. *border* – the resulting color is specified by the *borderColor* element (RGBA)

In wrap mode *mirror*, the texture image is repeated both in horizontal and in vertical direction to fill the texture space similar to wrap mode *wrap*. Unlike *wrap*, each repetition results from flipping the previous texture part along the repetition direction. This behaviour removes the edge correspondence constraint for wrapped textures and always results in a seamless texture.

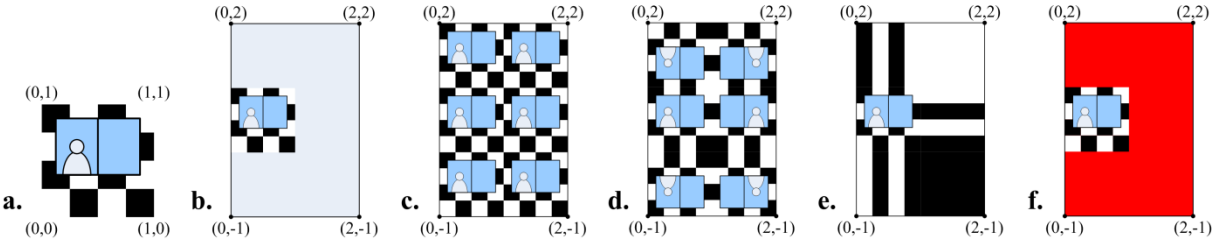

Fig. 15: A texture (a) applied to a facade using different wrap modes: (b) none, (c) wrap, (d) mirror, (e) clamp and (f) border. The border color is red. The numbers denote texture coordinates (image: Hasso-Plattner-Institute).

AbstractTextureType, _Texture, WrapModeType, TextureTypeType

```
<xs:complexType name="AbstractTextureType" abstract="true">
  <xs:complexContent>
    <xs:extension base="AbstractSurfaceDataType">
      <xs:sequence>
        <xs:element name="imageURI" type="xs:anyURI"/>
        <xs:element name="mimeType" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="textureType" type="TextureTypeType" minOccurs="0"/>
        <xs:element name="wrapMode" type="WrapModeType" minOccurs="0"/>
        <xs:element name="borderColor" type="ColorPlusOpacity" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfTexture" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_Texture" type="AbstractTextureType" abstract="true" substitutionGroup="_SurfaceData"/>
<!-- ========================================================================================== -->
```

```
<xs:element name="_GenericApplicationPropertyOfTexture" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:simpleType name="WrapModeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="wrap"/>
    <xs:enumeration value="mirror"/>
    <xs:enumeration value="clamp"/>
    <xs:enumeration value="border"/>
  </xs:restriction>
</xs:simpleType>
<!-- ========================================================================================== -->
<xs:simpleType name="TextureTypeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="specific"/>
    <xs:enumeration value="typical"/>
    <xs:enumeration value="unknown"/>
  </xs:restriction>
</xs:simpleType>
```

_Texture is further specialised according to the texture parameterisation, i.e. the mapping function from a location on the surface to a location in the texture image. CityGML uses the notion of texture space, where the texture image always occupies the region $[0,1]^2$ regardless of the actual image size or aspect ratio. The lower left image corner is located at the origin (some graphics APIs may use other conventions and require texture coordinate conversion). The mapping function must be known for each surface geometry object to receive texture.

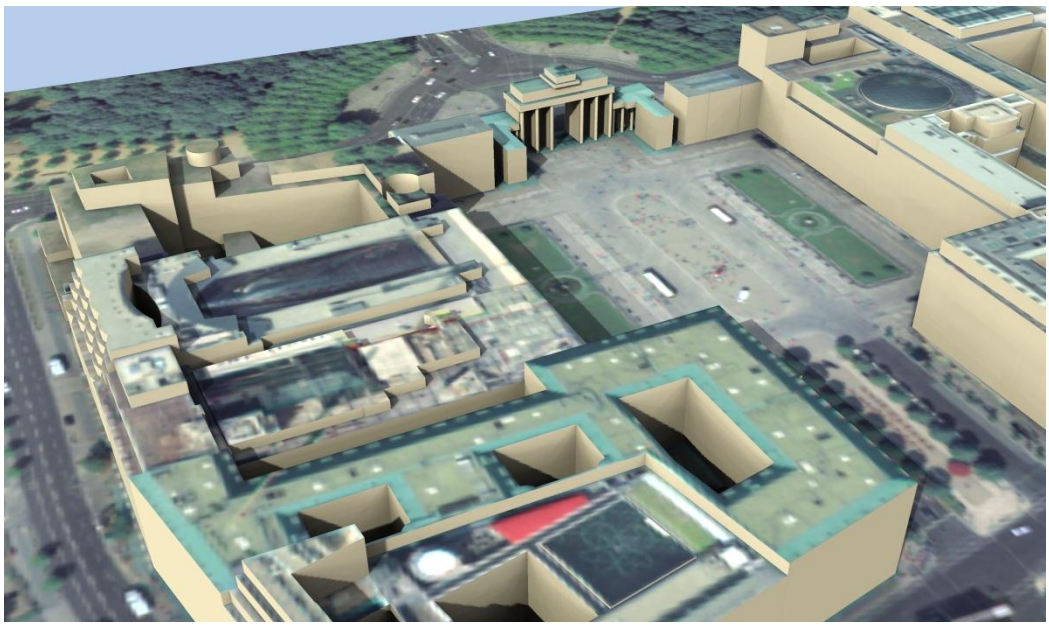

Fig. 16: A georeferenced texture applied to ground and roof surfaces (source: Senate of Berlin, Hasso-Plattner-Institute).

The class *GeoreferencedTexture* describes a texture that uses a planimetric projection. Consequently, it does not make sense to texture vertical surfaces using a *GeoreferencedTexture*. Such a texture has a unique mapping function which is usually provided with the image file (e.g. georeferenced TIFF) or as a separate ESRI world file[1]. The search order for an external georeference is determined by the boolean flag *preferWorldFile*. If this flag is set to true (its default value), a world file is looked for first and only if it is not found the georeference from the image data is used. If *preferWorldFile* is false, the world file is used only if no georeference from the image data is available.

Alternatively, CityGML allows for inline specification of a georeference similar to a world file. This internal georeference specification always takes precedence over any external georeference. *referencePoint* defines the location of the center of the upper left image pixel in world space and corresponds to values 5 and 6 in an ESRI world file. Since *GeoreferencedTexture* uses a planimetric projection, *referencePoint* is two-dimensional. *orientation* defines the rotation and scaling of the image in form of a 2x2 matrix (a list of 4 doubles in row-major order corresponding to values 1, 3, 2, and 4 in an ESRI world file). The CRS of this transformation is identical to

[1] Further information about the ESRI world file format is provided at http://en.wikipedia.org/wiki/World_file.

the *referencePoint*'s CRS. A planimetric point $(x,y)^T$ in that CRS is transformed to a point $(s,t)^T$ in texture space using the formula:

$$\begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} 1/w & 0 \\ 0 & -1/h \end{pmatrix} \cdot M^{-1} \cdot \left(\begin{pmatrix} x \\ y \end{pmatrix} - P_R \right) + \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

with *M* denoting *orientation*, P_R denoting *referencePoint*., *w* the image's width in pixels, and *h* the image's height in pixels. This transformation compensates for the difference between the image coordinate system used in ESRI world files (origin in upper left corner, positive x-axis rightwards, and positive y-axis downwards) and texture space in CityGML (origin in lower left corner, positive x-axis rightwards, and positive y-axis upwards).

If neither an internal nor an external georeference is given the *GeoreferencedTexture* is invalid. Each target surface geometry object is specified by an URI in a *target* element. All target surface geometry objects share the mapping function defined by the georeference. No other mapping function is allowed. Please note, that the *gml:boundedBy* property inherited from *gml:AbstractFeatureType* could be set to the bounding box of valid image data to allow for spatial queries. Fig. 16 shows a georeferenced texture applied to the ground and all roof surfaces.

GeoreferencedTextureType, GeoreferencedTexture

```
<xs:complexType name="GeoreferencedTextureType">
  <xs:complexContent>
    <xs:extension base="AbstractTextureType">
      <xs:sequence>
        <xs:element name="preferWorldFile" type="xs:boolean" default="true" minOccurs="0"/>
        <xs:element name="referencePoint" type="gml:PointPropertyType" minOccurs="0"/>
        <xs:element name="orientation" type="core:TransformationMatrix2x2Type" minOccurs="0"/>
        <xs:element name="target" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfGeoreferencedTexture" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="GeoreferencedTexture" type="GeoreferencedTextureType" substitutionGroup="_Texture"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfGeoreferencedTexture" type="xs:anyType" abstract="true"/>
```

The class *ParameterizedTexture* describes a texture with target-dependent mapping function. The mapping is defined by subclasses of class *_TextureParameterization* as a property of the link to the target surface geometry object. Each target surface geometry object is specified as URI in the *uri* attribute of a separate *target* element. Since *target* implements *gml:AssociationAttributeGroup*, it allows referencing to a remote *_TextureParameterization* object (using the *xlink:href* attribute), e.g. for sharing a mapping function between targets or textures in different themes. The mapping function can either use the concept of texture coordinates (through class *TexCoordList*) or a transformation matrix from world space to texture space (through class *TexCoordGen*).

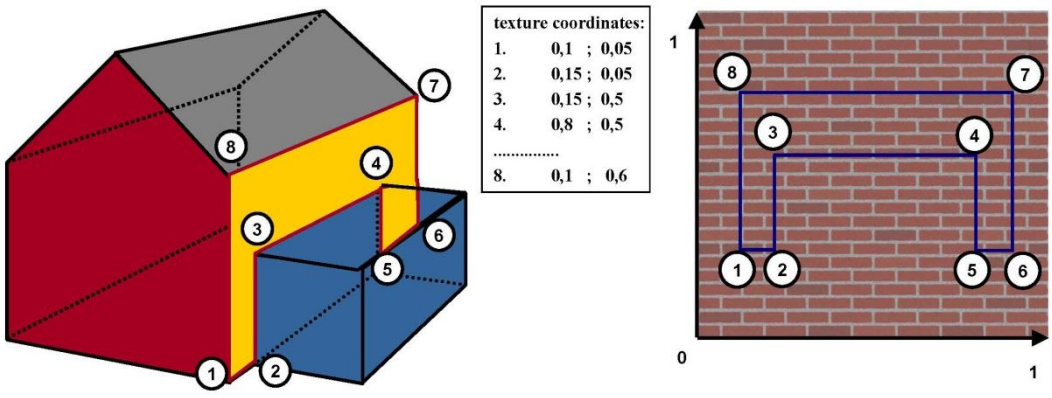

Fig. 17: Positioning of textures using texture coordinates (image: IGG Uni Bonn).

Texture coordinates are applicable only to polygonal surfaces, whose boundaries are described by *gml:LinearRing* (e.g., *gml:Triangle*, *gml:Polygon*, or a *gml:MultiSurface* consisting of *gml:Polygon*s). They define an explicit mapping of a surface's vertices to points in texture space, i.e. each vertex including interior ring vertices must receive a corresponding coordinate pair in texture space (for the notion of coordinates, refer to ISO 19111). These coordinates are not restricted to the [0,1] interval. Texture coordinates for interior surface points are planarly interpolated from the vertices' texture coordinates. Fig. 16 shows an example.

Texture coordinates for a target surface geometry object are specified using class *TexCoordList* as a texture parameterization object in the texture's *target* property. Each exterior and interior *gml:LinearRing* composing the boundary of the target surface geometry object (which also might be a *gml:CompositeSurface*, *gml:MultiSurface*, or *gml:TriangulatedSurface*) requires its own set of texture coordinates. A set of texture coordinates is specified using the *textureCoordinates* element of class *TexCoordList*. Thus, a *TexCoordList* contains as many *textureCoordinate* elements as the target surface geometry object contains *gml:LinearRing*s. *textureCoordinate*'s mandatory attribute *ring* provides the *gml:id* of the respective ring. The content is an ordered list of double values where each two values define a $(s,t)^T$ texture coordinate pair with *s* denoting the horizontal and *t* the vertical texture axis. The list contains one pair per ring point with the pairs' order corresponding to the ring points' order in the CityGML document (regardless of a possibly flipped surface orientation). If any ring point of a target surface geometry object has no texture coordinates assigned, the mapping is incomplete and the respective surface cannot be textured. In case of aggregated target geometry objects, mapping completeness is determined only for leaf geometry objects.

a.  b. 

Fig. 18: Projecting a photograph (a) onto multiple facades (b) using the *worldToTexture* transformation. The photograph does not cover the left facade completely. Thus, the texture appears to be clipped. Texture wrapping is set to "*none*" (source: Senate of Berlin, Hasso-Plattner-Institute).

Alternatively, the mapping function can comprise a 3x4 transformation matrix specified by class *TexCoordGen*. The transformation matrix, specified by the *worldToTexture* element, defines a linear transformation from a spatial location in homogeneous coordinates to texture space. The use of homogeneous coordinates facilitates perspective projections as transformation, e.g. for projecting a photograph into a city model (cf. Fig. 18). Texture coordinates $(s,t)^T$ are calculated from a space location $(x,y,z)^T$ as $(s,t)^T = (s'/q', t'/q')^T$ with $(s',t',q')^T = M \cdot (x,y,z,1)^T$. *M* denotes the 3x4 transformation matrix. Compared to a general 4x4 transformation, the resulting *z* component is ignored. Thus, the respective matrix row is omitted. Additionally, the *worldToTexture* element uses the *gml:SRSReferenceGroup* attributes to define its CRS. A location in world space has to be first transformed into this CRS before the transformation matrix can be applied.

The following construction results in a *worldToTexture* transformation that mimics the process of taking a photograph by projecting a location in world space (in the city model) to a location in texture space:

$$M = \underbrace{\begin{pmatrix} 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{pmatrix}}_{\substack{\text{Adjustment to} \\ \text{texture space}}} \cdot \underbrace{\begin{pmatrix} 2f/w & 0 & 0 & 0 \\ 0 & 2f/h & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\text{Perspective projection}} \cdot \underbrace{\begin{pmatrix} r_x & r_y & r_z & 0 \\ u_x & u_y & u_z & 0 \\ d_x & d_y & d_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{Camera orientation}} \cdot \underbrace{\begin{pmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{Camera location}}$$

In this formula, *f* denotes the focal length; *w* and *h* represent the image sensor's physical dimensions; $\vec{r}$, $\vec{u}$, and $\vec{d}$ define the camera's frame of reference as right, up and directional unit vectors expressed in world coordinates; and *P* stands for the camera's location in world space. Fig. 19 sketches this setting.

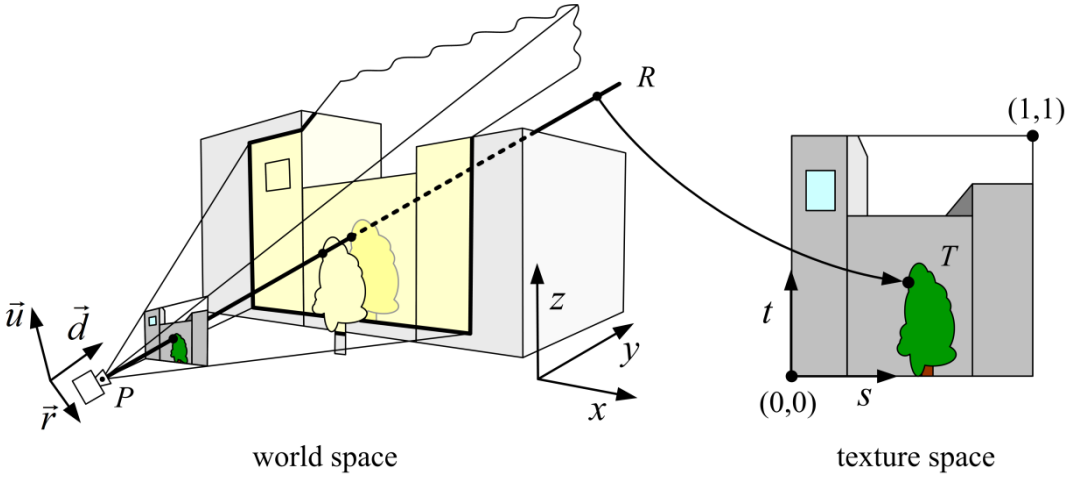

Fig. 19: Projective texture mapping. All points on a ray *R* starting from the projection center *P* are mapped to the same point *T* in texture space (image: Hasso-Plattner-Institute, IGG TU Berlin).

Alternatively, if the 3x4 camera matrix M_P is known (e.g. through a calibration and registration process), it can easily be adopted for use in *worldToTexture*. M_P is derived from intrinsic and extrinsic camera parameters (interior and exterior orientation) and transforms a location in world space to a pixel location in the image. Assuming the upper left image corner has pixel coordinates (0,0), the complete transformation to texture space coordinates can be written as ($width_{image}$ and $height_{image}$ denote the image size in pixels):

$$M = \begin{pmatrix} 1/width_{image} & 0 & 0 \\ 0 & -1/height_{image} & 1 \\ 0 & 0 & 1 \end{pmatrix} \cdot M_P$$

Please note, that *worldToTexture* cannot compensate for radial or other non-linear distortions introduced by a real camera lens.

Another use of *worldToTexture* is texturing a facade with complex geometry without specifying texture coordinates for each *gml:LinearRing*. Instead, only the facade's aggregated surface becomes the texture target using a *TexCoordGen* as parameterization. Then, *worldToTexture* effectively encodes an orthographic projection of world space into texture space. For the special case of a vertical facade this transformation is given by:

$$M = \underbrace{\begin{pmatrix} 1/width_f & 0 & 0 & 0 \\ 0 & 1/height_f & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{Scaling\,to\,texture\,space} \cdot \underbrace{\begin{pmatrix} -n_y & n_x & 0 & 0 \\ 0 & 0 & 1 & 0 \\ n_x & n_y & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{Facade\,orientation} \cdot \underbrace{\begin{pmatrix} 1 & 0 & 0 & -F_x \\ 0 & 1 & 0 & -F_y \\ 0 & 0 & 1 & -F_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{Facade\,location}$$

This equation assumes $\vec{n}$ denoting the facade's overall normal vector (normalized, pointing outward, and being parallel to the ground), *F* denoting the facade's lower left point, and $width_f$ and $height_f$ specifying the facade's dimensions in world units. For the general case of an arbitrary normal vector the facade orientation matrix assumes a form similar to the camera orientation matrix:

$$M = \begin{pmatrix} 1/width_f & 0 & 0 & 0 \\ 0 & 1/height_f & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} r_x & r_y & r_z & 0 \\ u_x & u_y & u_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -F_x \\ 0 & 1 & 0 & -F_y \\ 0 & 0 & 1 & -F_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ with } \vec{r} = \frac{(0 \;\; 0 \;\; 1)^T \times \vec{n}}{\left\| (0 \;\; 0 \;\; 1)^T \times \vec{n} \right\|}, \quad \vec{u} = \vec{n} \times \vec{r}$$

ParameterizedTextureType, ParameterizedTexture, TextureAssociationType

```
<xs:complexType name="ParameterizedTextureType">
  <xs:complexContent>
    <xs:extension base="AbstractTextureType">
      <xs:sequence>
        <xs:element name="target" type="TextureAssociationType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfParameterizedTexture" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="ParameterizedTexture" type="ParameterizedTextureType" substitutionGroup="_Texture"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfParameterizedTexture" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="TextureAssociationType">
  <xs:sequence minOccurs="0">
    <xs:element ref="_TextureParameterization"/>
  </xs:sequence>
  <xs:attribute name="uri" type="xs:anyURI" use="required"/>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
```

AbstractTextureParameterizationType, TexCoordListType, TexCoordGenType

```
<xs:complexType name="AbstractTextureParameterizationType" abstract="true">
  <xs:complexContent>
    <xs:extension base="gml:AbstractGMLType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfTextureParameterization" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_TextureParameterization" type="AbstractTextureParameterizationType" abstract="true"
            substitutionGroup="gml:_GML"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTextureParameterization" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="TexCoordListType">
  <xs:complexContent>
    <xs:extension base="AbstractTextureParameterizationType">
      <xs:sequence>
        <xs:element name="textureCoordinates" maxOccurs="unbounded">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="gml:doubleList">
                <xs:attribute name="ring" type="xs:anyURI" use="required"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element ref="_GenericApplicationPropertyOfTexCoordList" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="TexCoordList" type="TexCoordListType" substitutionGroup="_TextureParameterization"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTexCoordList" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="TexCoordGenType">
  <xs:complexContent>
    <xs:extension base="AbstractTextureParameterizationType">
      <xs:sequence>
        <xs:element name="worldToTexture">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="core:TransformationMatrix3x4Type">
```

```
                    <xs:attributeGroup ref="gml:SRSReferenceGroup"/>
                  </xs:extension>
                </xs:simpleContent>
              </xs:complexType>
            </xs:element>
            <xs:element ref="_GenericApplicationPropertyOfTexCoordGen" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:element name="TexCoordGen" type="TexCoordGenType" substitutionGroup="_TextureParameterization"/>
    <!-- ========================================================================================== -->
    <xs:element name="_GenericApplicationPropertyOfTexCoordGen" type="xs:anyType" abstract="true"/>
```

9.5 Related concepts

The notion of appearance clearly relates to the generic coverage approach (cf. ISO 19123 and OGC Abstract specification, Topic 6). Surface data can be described as discrete or continuous coverage over a surface as two-dimensional domain with a specific mapping function. Such an implementation requires the extension of GML coverages (as of version 3.1) by suitable mapping functions and specialisation for valid domain and range sets. For reasons of simplicity and comprehensibility both in implementation and usage, CityGML does not follow this approach, but relies on textures and materials as well-known surface property descriptions from the field of computer graphics (cf. X3D, COLLADA specification, Foley et al.). Textures and materials store data as color using an appropriate mapping. If such a mapping is impractical, data storage can be customised using ADEs. A review of coverages for appearance modelling is considered for CityGML beyond version 2.0.0.

Appearance is also related to portrayal. Portrayal describes the composition and symbolisation of a digital model's image, i.e. presentation, while appearance encodes observations of the real object's surface, i.e. data. Even though being based on graphical terms such as textures and materials, surface data is not limited to being input for portrayal, but similarly serves as input or output for analyses on a feature's surface. Consequently, CityGML does not define mixing or composition of themes for portrayal purposes. Portrayal is left to viewer applications or styling specification languages such as OGC Styled Layer Descriptors (SLD) or OGC Symbology Encoding (SE).

9.6 Code lists

The *mimeType* attribute of the feature *_Texture* is specified as *gml:CodeType*. The values of this property can be enumerated in a code list. A proposal for this code list can be found in annex C.6.

9.7 Conformance requirements

Base requirements

1. A surface geometry object may be the target of at most two textures and two materials (one for front and back respectively) per theme.
2. The *referencePoint* property (type: *gml:PointPropertyType*) of the element *GeoreferencedTexture* may only contain or reference a point geometry object with 2D coordinate values.
3. Texture coordinates given by the *textureCoordinates* property of the element *TexCoordList* define an explicit mapping of a surface's boundary points to points in texture space. Each boundary point of the surface must receive a corresponding coordinate pair in texture space. The coordinate pair in texture space shall be given as two doubles per boundary point. The order of the coordinate pairs must follow the order of the boundary points in the CityGML document (regardless of a possibly flipped surface orientation). Each *gml:LinearRing* composing the boundary of the target surface geometry object requires its own set of texture coordinates.
4. A *GeoreferencedTexture* element must provide either internal or external georeference, otherwise it is invalid. Internal georeference shall be declared by the *referencePoint* property (type: *gml:PointPropertyType*) and the *orientation* property (type: *core:TransformationMatrix2x2Type*) of the

element *GeoreferencedTexture*. External georeference may be provided by the texture image file itself (e.g. GeoTIFF) or by an accompanying world file.

Referential integrity

5. The *appearanceMember* element (type: *AppearancePropertyType*) may contain an *Appearance* element inline or an XLink reference to a remote *Appearance* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *appearanceMember* element may only point to a remote *Appearance* element (where remote *Appearance* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
6. The *appearance* property (type: *AppearancePropertyType*) of the element *core:_CityObject* may contain an *Appearance* element inline or an XLink reference to a remote *Appearance* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *appearance* property may only point to a remote *Appearance* element (where remote *Appearance* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
7. The *surfaceDataMember* property (type: *SurfaceDataPropertyType*) of the element *Appearance* may contain a *_SurfaceData* element inline or an XLink reference to a remote *_SurfaceData* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *surfaceDataMember* property may only point to a remote *_SurfaceData* element (where remote *_SurfaceData* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
8. The *target* property (type: *TextureAssociationType*) of the element *ParameterizedTexture* may contain a *_TextureParameterization* element inline or an XLink reference to a remote *_ TextureParameterization* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *target* property may only point to a remote *_ TextureParameterization* element (where remote *_ TextureParameterization* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
9. The *target* property (type *xs:anyURI*) of the element *GeoreferencedTexture* shall specify the *gml:id* of the target surface geometry object which may only be of type *gml:AbstractSurfaceType* or *gml:MultiSurface*.
10. The *uri* attribute of the complex type *TextureAssociationType* shall specify the *gml:id* of the target surface geometry object which may only be of type *gml:AbstractSurfaceType* or *gml:MultiSurface*.
11. The *ring* attribute of the *textureCoordinates* property of the element *TexCoordList* shall specify the *gml:id* of the target surface geometry object which may only be of type *gml:LinearRing*.
12. The *target* property (type *xs:anyURI*) of the element *X3DMaterial* shall specify the *gml:id* of the target surface geometry object which may only be of type *gml:AbstractSurfaceType* or *gml:MultiSurface*.

9.8 Material model of previous CityGML versions [deprecated]

Since GML3 has no built-in concept for the representation of surface materials, previous versions of CityGML extend the GML3 geometry model by the class *TexturedSurface*, which allows for assigning appearance properties (colors, shininess, transparency) and textures to 3D surfaces. The definition of the appearance properties is adopted from the X3D specification. This approach for appearance modelling has been *deprecated* due to inherent limitations. However, in order to provide a certain degree of backwards compatibility for already existing CityGML implementations, the approach has been incorporated into CityGML version 1.0 and version 2.0 as a separate extension module called *TexturedSurface*. By this means, implementations may employ the old material model by supporting this module. Please note, that appearance information modelled according to the *TexturedSurface* module can be converted without information loss to the concepts provided by CityGML's *Appearance* module that has been introduced in the previous clauses of this chapter. Thus, the use of the *TexturedSurface* module is *strongly discouraged* and implementations should only stick to the *Appearance* module instead. Moreover, the *TexturedSurface* module is expected to be removed in future versions of CityGML.

For the *TexturedSurface* module, each surface or composite surface can be specialized to a *TexturedSurface*, which can be assigned *Materials* (*colors, shininess, transparency*) or *SimpleTextures*. Fig. 20 depicts the UML diagram, for XML schema definition see annex A.14.

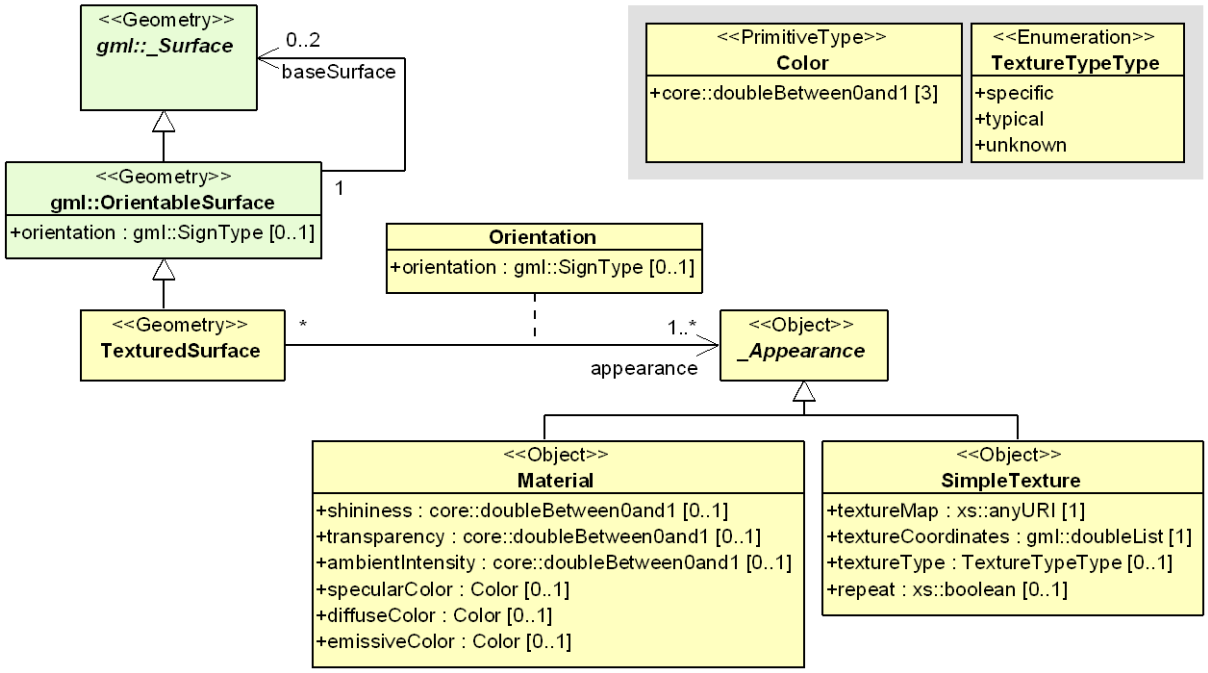

Fig. 20: UML diagram of CityGML's material model. Please note, that this approach for appearance modelling has been marked as deprecated and is expected to be removed in future CityGML versions. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *TexturedSurface* module.

The concept of positioning textures on surfaces complies with the 3D computer graphics standard X3D (web 3D 2004), a successor of VRML97. CityGML adds the class *TexturedSurface* to the geometry model of GML3 because there has been no appropriate texturing concept in ISO 19107 and in GML3.

A texture is specified as a raster image referenced by an *URI* (*Uniform Resource Identifier*) and can be an arbitrary resource, even on the internet. Textures are positioned by employing the concept of *texture coordinates*, i.e. each texture coordinate matches with exactly one 3D coordinate of the *TexturedSurface* (Fig. 17). The use of texture coordinates allows an exact positioning and trimming of the texture on the surface geometry.

The color of a surface is defined by RGB values. These have to be in the range of 0 to 1. The *frontOpacity* and the *backOpacity* define the level of *transparency* of each surface. Their values have also to be in the range of 0 to 1, where 1 means completely covering and 0 denotes a completely transparent surface. The colors can be differentiated in *diffuseColor* (color when illuminated by a source of light), *emissiveColor* (color when self-illuminating) and *specularColor*/*shininess* (color for shiny surfaces).

Textures can be qualified by the attribute *textureType*. The *textureType* differentiates between textures which are specific for a certain object (*specific*) and prototypic textures being typical for that object surface (*typical)*. Textures may also be classified as *unknown*.

_Appearance is derived from *gml:AbstractGMLType* to be referenced in an *appearance* property. The attribute *gml:id* is inherited, whose value may be referenced by a XLink. *_Appearance* is the parent class of *Material* and *SimpleTexture*.

XML namespace

The XML namespace of the CityGML *TexturedSurface* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/texturedsurface/2.0. Within the XML Schema definition of the *Textured-Surface* module, this URI is also used to identify the default namespace.

9.8.1 Textured surfaces

TexturedSurfaceType, TexturedSurface, AppearancePropertyType

```
<xs:complexType name="TexturedSurfaceType">
  <xs:complexContent>
    <xs:extension base="gml:OrientableSurfaceType">
      <xs:sequence>
        <xs:element ref="appearance" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="TexturedSurface" type="TexturedSurfaceType" substitutionGroup="gml:OrientableSurface"/>
<!-- ========================================================================================== -->
<xs:element name="appearance" type="AppearancePropertyType"/>
<!-- ========================================================================================== -->
<xs:complexType name="AppearancePropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="_Appearance"/>
  </xs:sequence>
  <xs:attribute name="orientation" type="gml:SignType" default="+"/>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
```

TexturedSurface may have one or more appearance properties, which can either be a *Material* (Color,...) or a *SimpleTexture*. The *_Appearance* element can either be represented inline as an element of this type or by an XLink reference to a remote *_Appearance* element. Either the reference or the contained element must be given, but neither both nor none. The side of the surface the *_Appearance* refers to is given by the *orientation* attribute (type *gml:SignType*) of the appearance property element, which refers to the corresponding *orientation* attribute of the orientable surface: + means the side with positive orientation and - the side with negative orientation.

AbstractAppearanceType, _Appearance

```
<xs:complexType name="AbstractAppearanceType" abstract="true">
  <xs:complexContent>
    <xs:extension base="gml:AbstractGMLType"/>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_Appearance" type="AbstractAppearanceType" abstract="true" substitutionGroup="gml:_GML"/>
```

MaterialType, Material

```
<xs:complexType name="MaterialType">
  <xs:complexContent>
    <xs:extension base="AbstractAppearanceType">
      <xs:sequence>
        <xs:element name="shininess" type="core:doubleBetween0and1" minOccurs="0"/>
        <xs:element name="transparency" type="core:doubleBetween0and1" minOccurs="0"/>
```



```
        <xs:element name="ambientIntensity" type="core:doubleBetween0and1" minOccurs="0"/>
        <xs:element name="specularColor" type="Color" minOccurs="0"/>
        <xs:element name="diffuseColor" type="Color" minOccurs="0"/>
        <xs:element name="emissiveColor" type="Color" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="Material" type="MaterialType" substitutionGroup="_Appearance"/>
```

SimpleTextureType, SimpleTexture, TextureTypeType

```
<xs:complexType name="SimpleTextureType">
  <xs:complexContent>
    <xs:extension base="AbstractAppearanceType">
      <xs:sequence>
        <xs:element name="textureMap" type="xs:anyURI"/>
        <xs:element name="textureCoordinates" type="gml:doubleList"/>
        <xs:element name="textureType" type="TextureTypeType" minOccurs="0"/>
        <xs:element name="repeat" type="xs:boolean" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="SimpleTexture" type="SimpleTextureType" substitutionGroup="_Appearance"/>
<!-- ========================================================================================= -->
<xs:simpleType name="TextureTypeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="specific"/>
    <xs:enumeration value="typical"/>
    <xs:enumeration value="unknown"/>
  </xs:restriction>
</xs:simpleType>
```

9.8.2 Conformance requirements

Referential integrity

- The *appearance* property (type: *AppearancePropertyType*) of the element *TexturedSurface* may contain an *_Appearance* element inline or an XLink reference to a remote *_Appearance* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *appearance* property may only point to a remote *_Appearance* element (where remote *_Appearance* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

10 Thematic model

The thematic model of CityGML consists of the class definitions for the most important types of objects within virtual 3D city models. These classes have been identified to be either required or important in many different application areas. Most thematic classes are (transitively) derived from the basic classes *_Feature* and *_FeatureCollection*, the basic notions defined in ISO 19109 and GML3 for the representation of spatial objects and their aggregations. Features contain spatial as well as non-spatial attributes which are mapped to GML3 feature properties with corresponding data types. Geometric properties are represented as associations to the geometry classes described in chapter 8. The thematic model also comprises different types of interrelationships between feature classes like aggregations, generalisations and associations.

The aim of the explicit modelling is to reach a high degree of semantic interoperability between different applications. By specifying the thematic concepts and their semantics along with their mapping to UML and GML3 different applications can rely on a well-defined set of *feature types*, attributes and data types with a standardised meaning or interpretation. In order to also allow for the exchange of objects and/or attributes that are not explicitly modelled in CityGML, the concepts of *generic city objects* and *attributes* as well as CityGML's *Application Domain Extension* mechanism have been introduced (cf. chapter 10.12 and chapter 10.13).

Each field of CityGML's thematic model is covered by a separate CityGML extension module. Thus, the extension modules are derived by vertically slicing the overall thematic data model of CityGML. All extension modules are based on and are dependent from the CityGML core module. The core comprises the basic concepts and components of the CityGML data model. Implementations may choose to combine CityGML extension modules in conjunction with the core according to their specific information needs or application domain. As for version 2.0 of CityGML, the following thirteen thematic extension modules are defined: *Appearance*, *Bridge*, *Building*, *CityFurniture*, *CityObjectGroup*, *Generics*, *LandUse*, *Relief*, *Transportation*, *Tunnel*, *Vegetation*, *WaterBody*, and *TexturedSurface [deprecated]*. Valid combinations of CityGML modules are called CityGML profiles. By this means, CityGML profiles explicitly allow for partial implementations of the overall CityGML data model (cf. chapter 7).

The thematic fields covered by the CityGML data model are introduced within the sub clauses of this chapter. Each sub clause is related to a specific CityGML module.

10.1 CityGML Core

The *CityGML Core* module defines the basic concepts and components of the overall CityGML data model. It forms the universal lower bound of the CityGML data model and, thus, is a dependency of all extension modules. Consequently, the core module has to be implemented by any conformant system. Primarily, the core module provides the abstract base classes from which thematic classes within extension modules are (transitively) derived. Besides abstract type definitions, the core also contains non-abstract content, for example basic data types and thematic classes that may be used by more than one extension module. The UML diagram in Fig. 21 illustrates CityGML's core module, for the XML Schema definition see below and annex A.1.

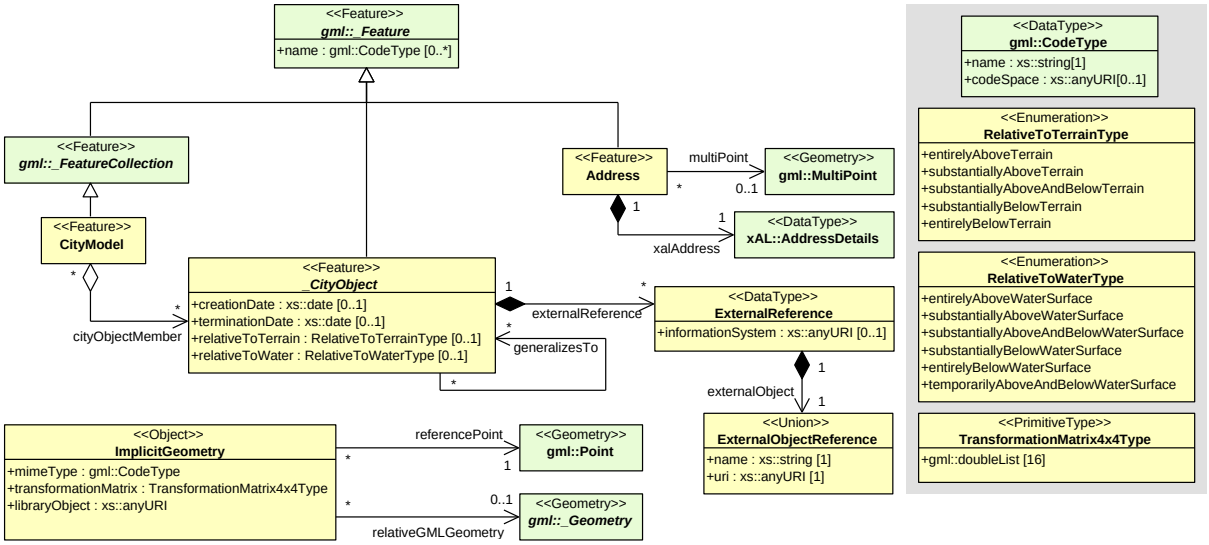

Fig. 21: UML diagram of CityGML's core module. The bracketed numbers following the attribute names denote the attribute's multiplicity: the minimal and maximal number of occurrences of the attribute per object. For example, a *name* is optional (0) in the class *_Feature* or may occur multiple times (star symbol), while a *_CityObject* has none or at most one *creationDate*. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the *CityGML Core* module.

The base class of all thematic classes within CityGML's data model is the abstract class *_CityObject*. *_CityObject* provides a creation and a termination date for the management of histories of features as well as the possibility to model external references to the same object in other data sets. Furthermore, two qualitative attributes *relativeToTerrain* and *relativeToWater* are provided which enable to specify the feature's location with respect to the terrain and water surface. The possible topological relations are illustrated in Fig. 22. Both attributes facilitate simple and efficient queries like for the number of subsurface buildings (*entirelyBelowTerrain*) without the need for an additional digital terrain model or a model of the water body.

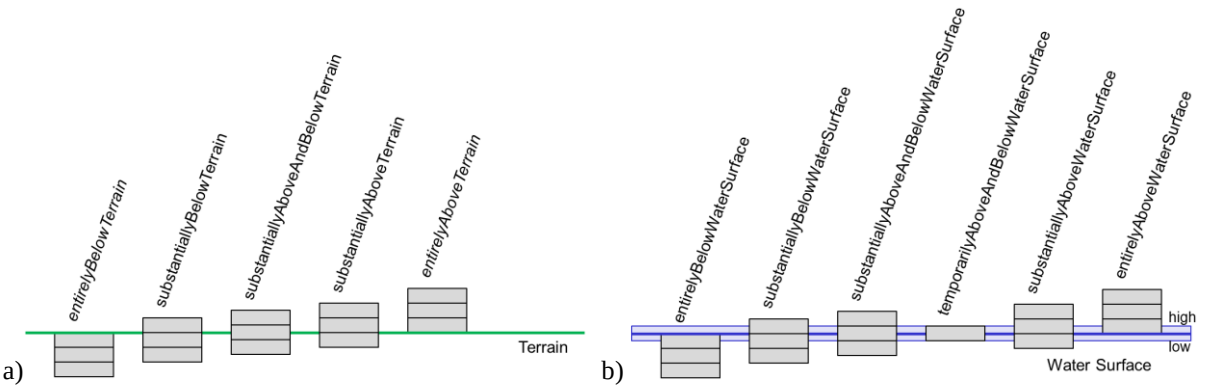

Fig. 22: Topological relations of a CityGML object with respect to a) the terrain and b) the water surface.

_CityObject is a subclass of the GML class *_Feature*, thus it inherits the metadata property (which can be e.g. information about the lineage, quality aspects, accuracy, local CRS) and name property from the superclass *_GML*. A *_CityObject* may have multiple names, which are optionally qualified by a *codeSpace*. This enables the differentiation between, for example, an official name and a popular name or of names in different languages

(cf. the name property of GML objects, Cox et al. 2004). The generalisation property *generalizesTo* of *_CityObject* may be used to relate features, which represent the same real-world object in different Levels-of-Detail, i.e. a feature and its generalised counterpart(s). The direction of this relation is from the feature to the corresponding generalised feature.

Thematic classes may have further subclasses with relations, attributes and geometry. Features of the specialized subclasses of *_CityObject* may be aggregated to a single *CityModel*, which is a feature collection with optional metadata. Generally, each feature has the attributes *class*, *function*, and *usage*, unless it is stated otherwise. The *class* attribute can occur only once, while the attributes *usage* and *function* can be used multiple times. The *class* attribute allows for the classification of features beyond the thematic class hierarchy of *_CityObject*. For example, a building feature is represented by the thematic subclass *bldg:Building* of *_CityObject* in the first place (this subclass is defined within CityGML's *Building* module, cf. chapter 10.3). A further classification, e.g. as residential or administration building, may then be modelled using the *class* attribute of the class *bldg:Building*. The attribute *function* normally denotes the intended purpose or usage of the object, such as hotel or shopping centre for a building, while the attribute *usage* normally defines its real or actual usage. Possible values for the attributes *class*, *function*, and *usage* can be specified in code lists which are recommended to be implemented as simple dictionaries following the *Simple Dictionary Profile* of GML 3.1.1 (cf. chapter 6.6 and 10.14). Annex C provides code lists proposed and maintained by the SIG 3D which contain feasible attribute values and which may be extended or redefined by users.

In addition to thematic content, the core module also provides the concept of implicit geometries as an enhancement of the geometry model of GML3. Since this concept is strongly related to the spatial model of CityGML it has already been introduced in chapter 8.2.

The top level class hierarchy of the thematic model in CityGML is presented in Fig. 23. The subclasses of *_CityObject* comprise the different thematic fields of a city model covered by separate CityGML extension modules: the terrain, buildings, bridges, tunnels, the coverage by land use objects, water bodies, vegetation, generic city objects, city furniture objects, city object groups, and transportation. To indicate the extension module defining a respective subclass of *_CityObject*, the class names in Fig. 23 are preceded by prefixes. Each prefix is associated with one CityGML extension module (see chapter 4.3 and chapter 7 for a list of CityGML's extension modules and the corresponding prefixes).

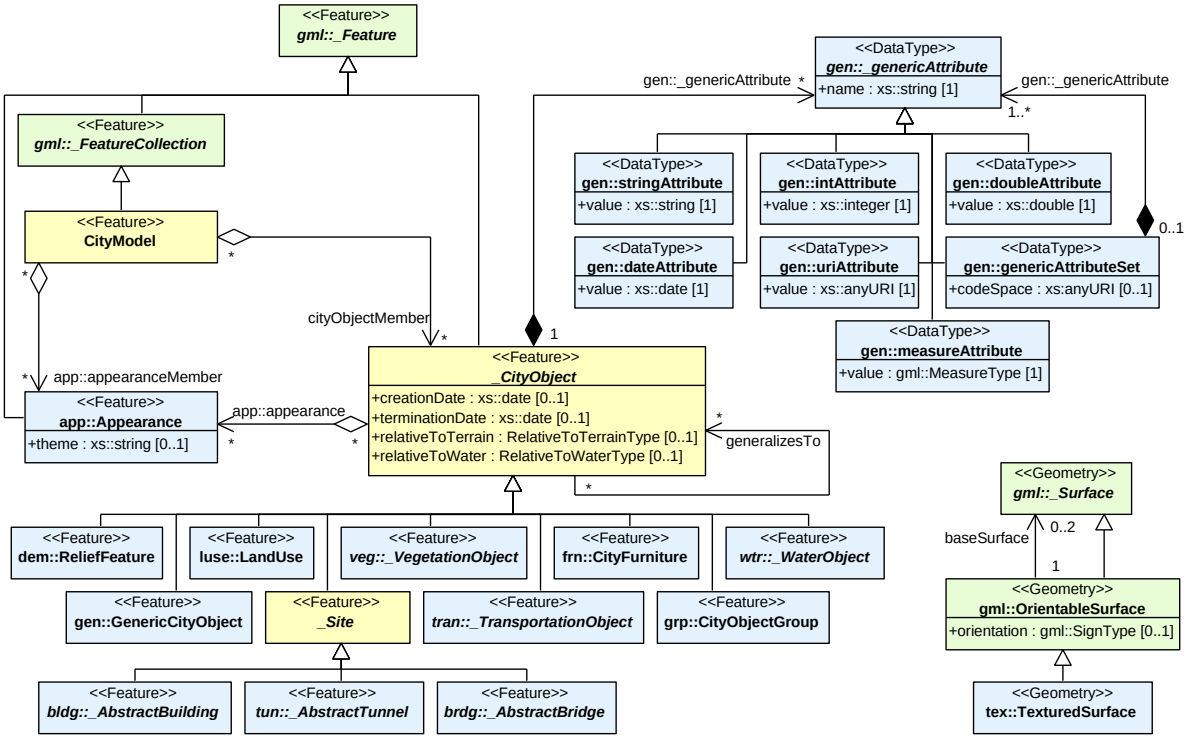

Fig. 23: CityGML's top level class hierarchy. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the *CityGML Core* module.

The classes *GenericCityObject* and *_genericAttribute* defined within CityGML's *Generics* module (cf. chapters 6.11 and 10.12) allow for modelling and exchanging of 3D objects which are not covered by any other thematic class or which require attributes not represented in CityGML. For example, in the future, sites derived from the abstract class *_Site* of the core module may be completed by further subclasses like excavation, city wall or embankment. At present, the class *GenericCityObject* should be used in order to represent and exchange these features. However, the concept of generic city objects and attributes may only be used if appropriate thematic classes or attributes are not provided by any other CityGML module.

If the *Generics* module is employed, each *CityObject* may be assigned an arbitrary number of generic attributes in order to represent additional properties of features. For this purpose, the *Generics* module augments the abstract base class *_CityObject* by the property element *_genericAttribute*. The additional property *_genericAttribute* is injected into *_CityObject* using CityGML's *Application Domain Extension* mechanism (cf. chapter 10.13). By this means, each thematic subclass of *_CityObject* inherits this property and, thus, the possibility to contain generic attributes. Therefore, the *Generics* module has a deliberate impact on all CityGML extension modules defining thematic subclasses of *_CityObject*.

Appearance information about a feature's surfaces can be represented by the class *Appearance* provided by CityGML's *Appearance* module (cf. chapter 9). In contrast to the other thematic extensions to the core, *Appearance* is not derived from *_CityObject* but from the GML class *_Feature*. *_CityObject* features and *Appearance* features may be embraced within a single *CityModel* feature collection in an arbitrary or even mixed sequence using the *cityObjectMember* and *appearanceMember* elements, both being members of the substitution group *gml:featureMember* (cf. chapter 9 and chapter 10.1.1). Furthermore, feature appearances may be stored inline the *_CityObject* itself. In order to enable city objects to store appearance information, the *Appearance* module augments the abstract base class *_CityObject* by the property element *appearance* using CityGML's *Application Domain Extension* mechanism (cf. chapter 10.13). Consequently, the *appearance* property is only available for *_CityObject* and its thematic subclasses if the *Appearance* module is supported. Therefore, like the *Generics* module, the *Appearance* module has a deliberate impact on any other extension module.

For sake of completeness, the class *TexturedSurface* is also illustrated in Fig. 23. This approach of appearance modelling of previous versions of CityGML has been deprecated and is expected to be removed in future CityGML versions. Since the information covered by *TexturedSurface* can be losslessly converted to the *Appearance* module, the use of *TexturedSurface* is *strongly discouraged*.

XML namespace

The XML namespace of the *CityGML Core* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/2.0. Within the XML Schema definition of the core module, this URI is also used to identify the default namespace.

10.1.1 Base elements

AbstractCityObjectType, _CityObject

```
<xs:complexType name="AbstractCityObjectType" abstract="true">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureType">
      <xs:sequence>
        <xs:element name="creationDate" type="xs:date" minOccurs="0"/>
        <xs:element name="terminationDate" type="xs:date" minOccurs="0"/>
        <xs:element name="externalReference" type="ExternalReferenceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="generalizesTo" type="GeneralizationRelationType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="relativeToTerrain" type="RelativeToTerrainType" minOccurs="0"/>
        <xs:element name="relativeToWater" type="RelativeToWaterType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfCityObject" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_CityObject" type="AbstractCityObjectType" abstract="true" substitutionGroup="gml:_Feature"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfCityObject" type="xs:anyType" abstract="true"/>
```

CityModelType, CityModel

```
<xs:complexType name="CityModelType">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureCollectionType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfCityModel" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="CityModel" type="CityModelType" substitutionGroup="gml:_FeatureCollection"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfCityModel" type="xs:anyType" abstract="true"/>
```

cityObjectMember

```
<xs:element name="cityObjectMember" type="gml:FeaturePropertyType" substitutionGroup="gml:featureMember"/>
```

AbstractSiteType, _Site

```
<xs:complexType name="AbstractSiteType" abstract="true">
  <xs:complexContent>
    <xs:extension base="AbstractCityObjectType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfSite" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_Site" type="AbstractSiteType" abstract="true" substitutionGroup="_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfSite" type="xs:anyType" abstract="true"/>
```

The abstract class *_Site* is intended to be the superclass for buildings, bridges, tunnels, facilities, etc. Future extension of CityGML (e.g. excavations, city walls or embankments) would be modelled as subclasses of *_Site*. As subclass of *_CityObject*, a *_Site* inherits all attributes and relations, in particular the id, names, external references, and generalisation relations.

10.1.2 Generalisation relation, RelativeToTerrainType and RelativeToWaterType

GeneralizationRelationType

```
<xs:complexType name="GeneralizationRelationType">
  <xs:sequence minOccurs="0">
    <xs:element ref="_CityObject"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
```

RelativeToTerrainType, RelativeToWaterType

```
<xs:simpleType name="RelativeToTerrainType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="entirelyAboveTerrain"/>
    <xs:enumeration value="substantiallyAboveTerrain"/>
    <xs:enumeration value="substantiallyAboveAndBelowTerrain"/>
    <xs:enumeration value="substantiallyBelowTerrain"/>
    <xs:enumeration value="entirelyBelowTerrain"/>
  </xs:restriction>
</xs:simpleType>
<!-- ========================================================================================== -->
<xs:simpleType name="RelativeToWaterType">
```

```
  <xs:restriction base="xs:string">
    <xs:enumeration value="entirelyAboveWaterSurface"/>
    <xs:enumeration value="substantiallyAboveWaterSurface"/>
    <xs:enumeration value="substantiallyAboveAndBelowWaterSurface"/>
    <xs:enumeration value="substantiallyBelowWaterSurface"/>
    <xs:enumeration value="entirelyBelowWaterSurface"/>
    <xs:enumeration value="temporarilyAboveAndBelowWaterSurface"/>
  </xs:restriction>
</xs:simpleType>
```

10.1.3 External references

An *ExternalReference* defines a hyperlink from a *_CityObject* to a corresponding object in another information system, for example in the German cadastre (ALKIS), the German topographic information system (ATKIS), or the OS MasterMap®. The reference consists of the name of the external information system, represented by an URI, and the reference of the external object, given either by a string or by an URI. If the *informationSystem* element is missing in the *ExternalReference*, the *ExternalObjectReference* must be an URI.

ExternalReferenceType, ExternalObjectReferenceType

```
<xs:complexType name="ExternalReferenceType">
  <xs:sequence>
    <xs:element name="informationSystem" type="xs:anyURI" minOccurs="0"/>
    <xs:element name="externalObject" type="ExternalObjectReferenceType"/>
  </xs:sequence>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:complexType name="ExternalObjectReferenceType">
  <xs:choice>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="uri" type="xs:anyURI"/>
  </xs:choice>
</xs:complexType>
```

10.1.4 Address information

The CityGML core module provides the means to represent address information of real-world features within virtual city models. Since not every real-world feature is assigned an address, a correspondent *address* property is not defined for the base class *_CityObject*, but has to be explicitly modelled for a thematic subclass. For example, the building model declares *address* properties for its classes *_AbstractBuilding* and *Door*. Both classes are referencing the corresponding data types of the core module to represent address information (cf. chapter 10.3).

Addresses are modelled as GML features having one *xalAddress* property and an optional *multiPoint* property. For example, for a building feature the *multiPoint* property allows for the specification of the exact positions of the building entrances that are associated with the corresponding address. The point coordinates can be 2D or 3D. Modelling addresses as features has the advantage that GML3's method of representing features by reference (using XLinks) can be applied. This means, that addresses might be bundled as an address *FeatureCollection* that is stored within an external file or that can be served by an external Web Feature Service. The address property elements within the CityGML file then would not contain the address information inline but only references to the corresponding external features.

The address information is specified using the *xAL address standard* issued by the OASIS consortium (OASIS 2003), which provides a generic schema for all kinds of international addresses. Therefore, child elements of the *xalAddress* property of *Address* have to be structured according to the OASIS xAL schema.

AddressPropertyType, AddressType, Address

```
<xs:complexType name="AddressPropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="Address"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
```



```
</xs:complexType>
<!-- ========================================================================================== -->
<xs:complexType name="AddressType">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureType">
      <xs:sequence>
        <xs:element name="xalAddress" type="xalAddressPropertyType"/>
        <xs:element name="multiPoint" type="gml:MultiPointPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfAddress" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Address" type="AddressType" substitutionGroup="gml:_Feature"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfAddress" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="xalAddressPropertyType">
  <xs:sequence>
    <xs:element ref="xAL:AddressDetails"/>
  </xs:sequence>
</xs:complexType>
```

The following two excerpts of a CityGML dataset contain examples for the representation of German and British addresses in xAL. The address information is attached to building objects (*bldg:Building*) according to the CityGML *Building* module (cf. chapter 10.3). Generally, if a CityGML instance document contains address information, the namespace prefix "xAL" should be declared in the root element and must refer to "urn:oasis:names:tc:ciq:xsdschema:xAL:2.0". An example showing a complete CityGML dataset including a building with an address element is provided in annex G.1.

```
<bldg:Building>
  …
  <bldg:address>
    <core:Address>
      <core:xalAddress>
        <!-- Bussardweg 7, 76356 Weingarten, Germany -->
        <xAL:AddressDetails>
          <xAL:Country>
            <xAL:CountryName>Germany</xAL:CountryName>
            <xAL:Locality Type="City">
              <xAL:LocalityName>Weingarten</xAL:LocalityName>
              <xAL:Thoroughfare Type="Street">
                <xAL:ThoroughfareNumber>7</xAL:ThoroughfareNumber>
                <xAL:ThoroughfareName>Bussardweg</xAL:ThoroughfareName>
              </xAL:Thoroughfare>
              <xAL:PostalCode>
                <xAL:PostalCodeNumber>76356</xAL:PostalCodeNumber>
              </xAL:PostalCode>
            </xAL:Locality>
          </xAL:Country>
        </xAL:AddressDetails>
      </core:xalAddress>
      </core:Address>
  </bldg:address>
</bldg:Building>
```

```
<bldg:Building>
  …
  <bldg:address>
    <core:Address>
      <core:xalAddress>
        <!-- 46 Brynmaer Road Battersea LONDON, SW11 4EW United Kingdom -->
        <!-- source: http://xml.coverpages.org/xnal.html -->
        <xAL:AddressDetails>
          <xAL:Country>
            <xAL:CountryName>United Kingdom</xAL:CountryName>
            <xAL:Locality Type="City">
              <xAL:LocalityName>LONDON</xAL:LocalityName>
              <xAL:DependentLocality Type="District">
```

```
                <xAL:DependentLocalityName>Battersea</xAL:DependentLocalityName>
                <xAL:Thoroughfare>
                  <xAL:ThoroughfareNumber>46</xAL:ThoroughfareNumber>
                  <xAL:ThoroughfareName>Brynmaer Road</xAL:ThoroughfareName>
                </xAL:Thoroughfare>
              </xAL:DependentLocality>
              <xAL:PostalCode>
                <xAL:PostalCodeNumber>SW11 4EW</xAL:PostalCodeNumber>
              </xAL:PostalCode>
            </xAL:Locality>
          </xAL:Country>
        </xAL:AddressDetails>
      </core:xalAddress>
    </core:Address>
  </bldg:address>
</bldg:Building>
```

10.1.5 Code lists

The *mimeType* attribute of *ImplicitGeometry* is specified as *gml:CodeType*. The values of this property can be enumerated in a code list. A proposal for this code list can be found in annex C.6.

10.1.6 Conformance requirements

Base requirements

1. The *CityModel* element (type: *CityModelType*, substitutionGroup: *gml:_FeatureCollection*) shall only contain *cityObjectMember* elements (type: *gml:FeaturePropertyType*), *app:appearanceMember* elements (type: *app:AppearancePropertyType*), and *gml:featureMember* elements (type: *gml:FeaturePropertyType*) as feature members.
2. The type *ExternalObjectReference* introduces the two elements *name* (type: *xs:string*) and *uri* (type: *xs:anyURI*). The external reference may be specified by either of them. However, if the *informationSystem* property element (type: *xs:anyURI*) of the type *ExternalReferenceType* is not provided, the *uri* element of *ExternalObjectReference* must be given.
3. In order to represent address information about a feature, the corresponding thematic class of the feature shall define a property of the type *AddressPropertyType*. Thus, for all CityGML extension modules only the type *AddressPropertyType* shall be used for elements providing address information.
4. Since the concept of implicit geometries (cf. chapter 8.2) is part of the *CityGML Core* module, the conformance requirements introduced for implicit geometries (cf. chapter 8.3.3) are part of the conformance requirements of the core.

Referential integrity

5. The *cityObjectMember* element (type: *gml:FeaturePropertyType*) may contain a *_CityObject* element, which typically is an object from a derived subclass like *bldg:Building*, inline or an XLink reference to a remote *_CityObject* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *cityObjectMember* element may only point to a remote *_CityObject* element (where remote *_CityObject* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
6. The type *AddressPropertyType* may contain an *Address* element inline or an XLink reference to a remote *Address* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the corresponding element of type *AddressPropertyType* may only point to a remote *Address* element (where remote *Address* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

10.2 Digital Terrain Model (DTM)

An essential part of a city model is the terrain. The Digital Terrain Model (DTM) of CityGML is provided by the thematic extension module *Relief* (cf. chapter 7). In CityGML, the terrain is represented by the class *ReliefFeature* in LOD 0-4 (Fig. 24 depicts the UML diagram, for the XML schema definition see annex A.9). A *ReliefFeature* consists of one or more entities of the class *ReliefComponent.* Its validity may be restricted to a certain area defined by an optional *validity extent polygon*. As *ReliefFeature* and *ReliefComponent* are derivatives of *_CityObject*, the corresponding attributes and relations are inherited. The class *ReliefFeature* is associated with different concepts of terrain representations which can coexist. The terrain may be specified as a regular raster or grid (*RasterRelief)*, as a TIN (Triangulated Irregular Network, *TINReflief*), by break lines (*BreaklineRelief)*, or by mass points *(MasspointRelie*f). The four types are implemented by the corresponding GML3 classes: grids by *gml:RectifiedGridCoverage*, break lines by *gml:MultiCurve*, mass points by *gml:MultiPoint* and TINs either by *gml:TriangulatedSurface* or by *gml:Tin*. In case of *gml:TriangulatedSurfaces*, the triangles are given explicitly while in case of *gml:Tin* only 3D points are represented, where the triangulation can be reconstructed by standard methods (Delaunay triangulation, cf. Okabe et al. 1992). Break lines are represented by 3D curves. Mass points are simply a set of 3D points.

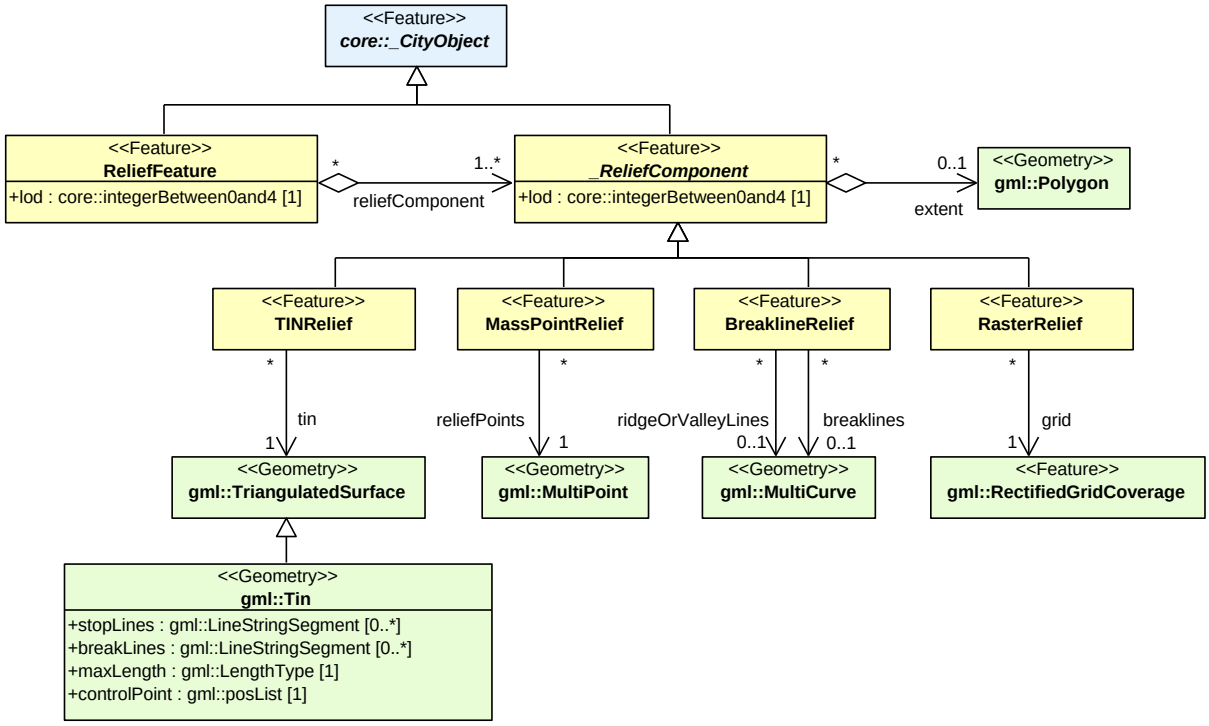

Fig. 24: UML diagram of the Digital Terrain Model in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Relief* module.

In a CityGML dataset the four terrain types may be combined in different ways, yielding a high flexibility. First, each type may be represented in different levels of detail, reflecting different accuracies or resolutions. Second, a part of the terrain can be described by the combination of multiple types, for example by a raster and break lines, or by a TIN and break lines. In this case, the break lines must share the geometry with the triangles. Third, neighboring regions may be represented by different types of terrain models. To facilitate this combination, each terrain object is provided with a spatial attribute denoting its *extent of validity* (Fig. 25). In most cases, the extent of validity of a regular raster dataset corresponds to its bounding box. This validity extent is represented by a 2D footprint polygon, which may have holes. This concept enables, for example, the modelling of a terrain by a coarse grid, where some distinguished regions are represented by a detailed, high-accuracy TIN. The boundaries between both types are given by the extent attributes of the corresponding terrain objects.

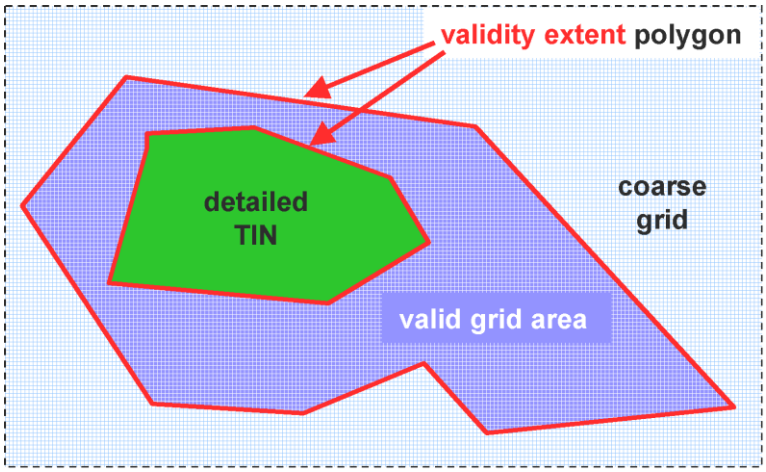

Fig. 25: Nested DTMs in CityGML using validity extent polygons (graphic: IGG Uni Bonn).

Accuracy and resolution of the DTM are not necessarily dependent on features of other CityGML extenstion modules such as building models. Hence, there is the possibility to integrate building models with higher LOD to a DTM with lower accuracy or resolution.

This approach interacts with the concept of *TerrainIntersectionCurves TIC* (cf. chapter 6.5). The TIC can be used like break lines to adjust the DTM to different features such as buildings, bridges, or city furnitures, and hence to ensure a consistent representation of the DTM. If necessary, a retriangulation may have to be processed. A TIC can also be derived by the individual intersection of the DTM and the corresponding feature.

ReliefFeature and its *ReliefComponents* both have an *lod* attribute denoting the corresponding level of detail. In most cases, the LOD of a *ReliefFeature* matches the LOD of its *ReliefComponents*. However, it is also allowed to specify a *ReliefFeature* with a high LOD which consists of *ReliefComponents* where some of them can have a LOD lower than that of the aggregating *ReliefFeature*. The idea is that, for example, for a LOD3 scene it might be sufficient to use a regular grid in LOD2 with certain higher precision areas defined by *ReliefComponents* in LOD3. The LOD2 grid and the LOD3 components can easily be integrated using the concept of the validity extent polygon. Therefore, although some of the *ReliefComponents* would have been classified to a lower LOD, the whole *ReliefFeature* would be appropriate to use with other LOD3 models which is indicated by setting its *lod* value to 3.

XML namespace

The XML namespace of the CityGML *Relief* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/relief/2.0. Within the XML Schema definition of the *Relief* module, this URI is also used to identify the default namespace.

10.2.1 Relief feature and relief component

ReliefFeatureType, ReliefFeature

```
<xs:complexType name="ReliefFeatureType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="lod" type="core:integerBetween0and4"/>
        <xs:element name="reliefComponent" type="ReliefComponentPropertyType" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfReliefFeature" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="ReliefFeature" type="ReliefFeatureType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfReliefFeature" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<xs:complexType name="ReliefComponentPropertyType">
```



```
  <xs:sequence minOccurs="0">
    <xs:element ref="_ReliefComponent"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
```

AbstractReliefComponentType, _ReliefComponent

```
<xs:complexType name="AbstractReliefComponentType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="lod" type="core:integerBetween0and4"/>
        <xs:element name="extent" type="gml:PolygonPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfReliefComponent" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_ReliefComponent" type="AbstractReliefComponentType" abstract="true" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfReliefComponent" type="xs:anyType" abstract="true"/>
```

10.2.2 TIN relief

TINReliefType, TINRelief

```
<xs:complexType name="TINReliefType">
  <xs:complexContent>
    <xs:extension base="AbstractReliefComponentType">
      <xs:sequence>
        <xs:element name="tin" type="tinPropertyType"/>
        <xs:element ref="_GenericApplicationPropertyOfTinRelief" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="TINRelief" type="TINReliefType" substitutionGroup="_ReliefComponent"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTinRelief" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="tinPropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="gml:TriangulatedSurface"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
```

The geometry of a *TINRelief* is defined by the GML geometry class *gml:TriangulatedSurface*. This allows either the explicit provision of a set of triangles (*gml:TriangulatedSurface*) or specifying of only the control points, break and stop lines using the subclass *gml:Tin* of *gml:TriangulatedSurface*. In the latter case, an application that processes an instance document containing a *gml:Tin* has to reconstruct the triangulated surface by the application of a constrained Delaunay triangulation algorithm (cf. Okabe et al. 1992).

10.2.3 Raster relief

RasterReliefType, RasterRelief, Elevation

```
<xs:complexType name="RasterReliefType">
  <xs:complexContent>
    <xs:extension base="AbstractReliefComponentType">
      <xs:sequence>
        <xs:element name="grid" type="gridPropertyType"/>
        <xs:element ref="_GenericApplicationPropertyOfRasterRelief" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="RasterRelief" type="RasterReliefType" substitutionGroup="_ReliefComponent"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfRasterRelief" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<!-- ========================================================================================= -->
<xs:complexType name="gridPropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="gml:RectifiedGridCoverage"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="Elevation" type="gml:LengthType" substitutionGroup="gml:_Object"/>
```

10.2.4 Mass point relief

MassPointReliefType, MassPointRelief

```
<xs:complexType name="MassPointReliefType">
  <xs:complexContent>
    <xs:extension base="AbstractReliefComponentType">
      <xs:sequence>
        <xs:element name="reliefPoints" type="gml:MultiPointPropertyType"/>
        <xs:element ref="_GenericApplicationPropertyOfMassPointRelief" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="MassPointRelief" type="MassPointReliefType" substitutionGroup="_ReliefComponent"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfMassPointRelief" type="xs:anyType" abstract="true"/>
```

10.2.5 Breakline relief

BreaklineReliefType, BreaklineRelief

```
<xs:complexType name="BreaklineReliefType">
  <xs:complexContent>
    <xs:extension base="AbstractReliefComponentType">
      <xs:sequence>
        <xs:element name="ridgeOrValleyLines" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="breaklines" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfBreaklineRelief" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="BreaklineRelief" type="BreaklineReliefType" substitutionGroup="_ReliefComponent"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfBreaklineRelief" type="xs:anyType" abstract="true"/>
```

The geometry of a *BreaklineRelief* can be composed of break lines and ridge/valley lines. Whereas break lines indicate abrupt changes of terrain slope, ridge/valley lines in addition mark a change of the sign of the terrain slope gradient. A *BreaklineRelief* must have at least one of the two properties.

10.2.6 Conformance requirements

Base requirements

1. The *gml:Polygon* geometry element describing the extent of validity of a *_ReliefComponent* element using the *extent* property (type: *gml:PolygonPropertyType*) of *_ReliefComponent* shall be given as 2D footprint polygon which may have inner holes.

Referential integrity

2. The *reliefComponent* property (type: *ReliefComponentPropertyType*) of the element *ReliefFeature* may contain a *_ReliefComponent* element inline or an XLink reference to a remote *_ReliefComponent* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *reliefComponent* property may only point to a remote *_ReliefComponent* element (where remote *_ReliefComponent* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
3. The *tin* property (type: *tinPropertyType*) of the element *TINRelief* may contain a *gml:TriangulatedSurface* element inline or an XLink reference to a remote *gml:TriangulatedSurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *tin* property may only point to a remote *gml:TriangulatedSurface* element (where remote *gml:TriangulatedSurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
4. The *grid* property (type: *gridPropertyType*) of the element *RasterRelief* may contain a *gml:RectifiedGridCoverage* element inline or an XLink reference to a remote *gml:RectifiedGridCoverage* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *grid* property may only point to a remote *gml:RectifiedGridCoverage* element (where remote *gml:RectifiedGridCoverage* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

10.3 Building model

The building model is one of the most detailed thematic concepts of CityGML. It allows for the representation of thematic and spatial aspects of buildings and building parts in five levels of detail, LOD0 to LOD4. The building model of CityGML is defined by the thematic extension module *Building* (cf. chapter 7). Fig. 26 provides examples of 3D city and building models in LOD1 – 4.

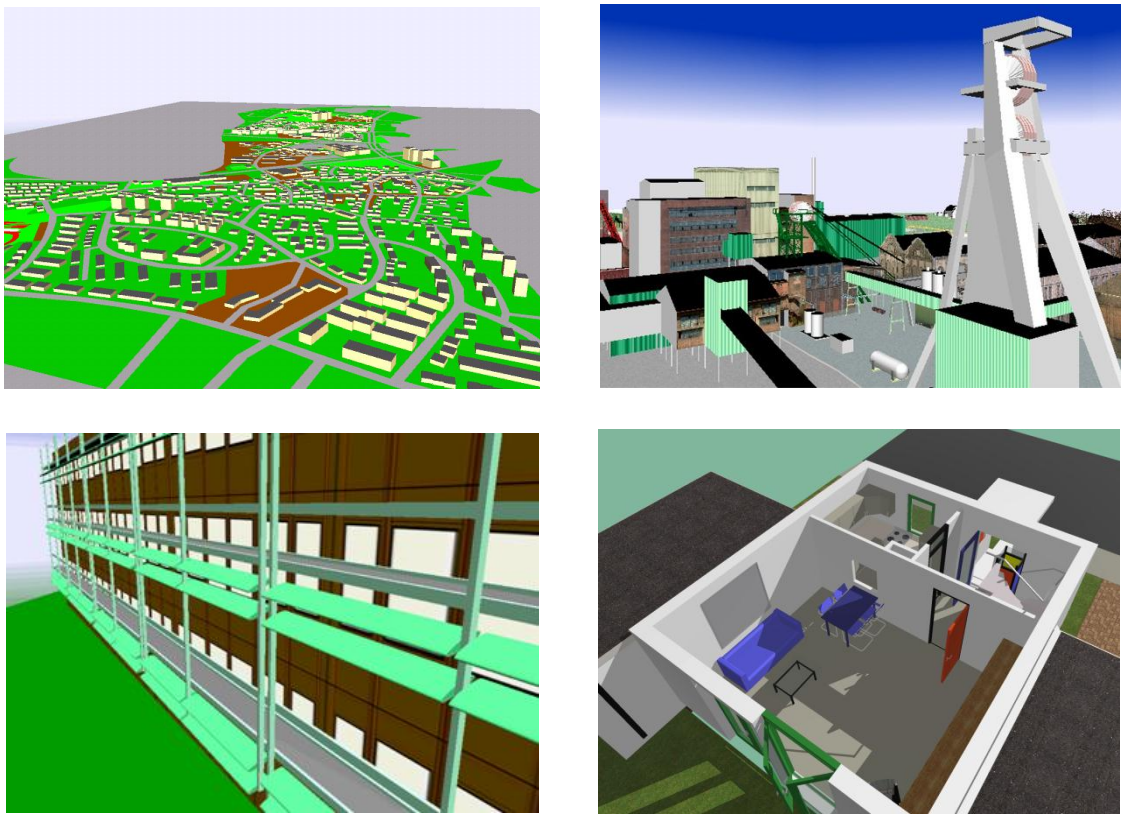

Fig. 26: Examples for city or building models in LOD1 (upper left), LOD2 (upper right), LOD3 (lower left), and LOD4 (lower right) (source: District of Recklinghausen, m-g-h ingenieure+architekten GmbH).

The UML diagram of the building model is depicted in Fig. 27, for the XML schema definition see annex A.4 and below. The pivotal class of the model is *_AbstractBuilding*, which is a subclass of the thematic class *_Site* (and transitively of the root class *_CityObject*). *_AbstractBuilding* is specialised either to a *Building* or to a *BuildingPart*. Since an *_AbstractBuilding* consists of *BuildingParts*, which again are *_AbstractBuildings*, an aggregation hierarchy of arbitrary depth may be realised. As subclass of the root class *_CityObject*, an *_AbstractBuilding* inherits all properties from *_CityObject* like the GML3 standard feature properties (*gml:name* etc.) and the CityGML specific properties like *ExternalReferences* (cf. chapter 6.7). Further properties not explicitly covered by *_AbstractBuilding* may be modelled as g*eneric attributes* provided by the CityGML *Generics* module (cf. chapter 10.12) or using the CityGML Application Domain Extension mechanism (cf. chapter 10.13).

Building complexes, which consist of a number of distinct buildings like a factory site or hospital complex, should be aggregated using the concept of *CityObjectGroups* (cf. chapter 6.8). The main building of the complex can be denoted by providing "main building" as the role name of the corresponding group member.

Both classes *Building* and *BuildingPart* inherit the attributes of *_AbstractBuilding*: the class of the building, the function (e.g. residential, public, or industry), the usage, the year of construction, the year of demolition, the roof type, the measured height, and the number and individual heights of the storeys above and below ground. This set of parameters is suited for roughly reconstructing the three-dimensional shape of a building and can be provided by cadastral systems. Furthermore, *Address* features can be assigned to *Buildings* or *BuildingParts*.

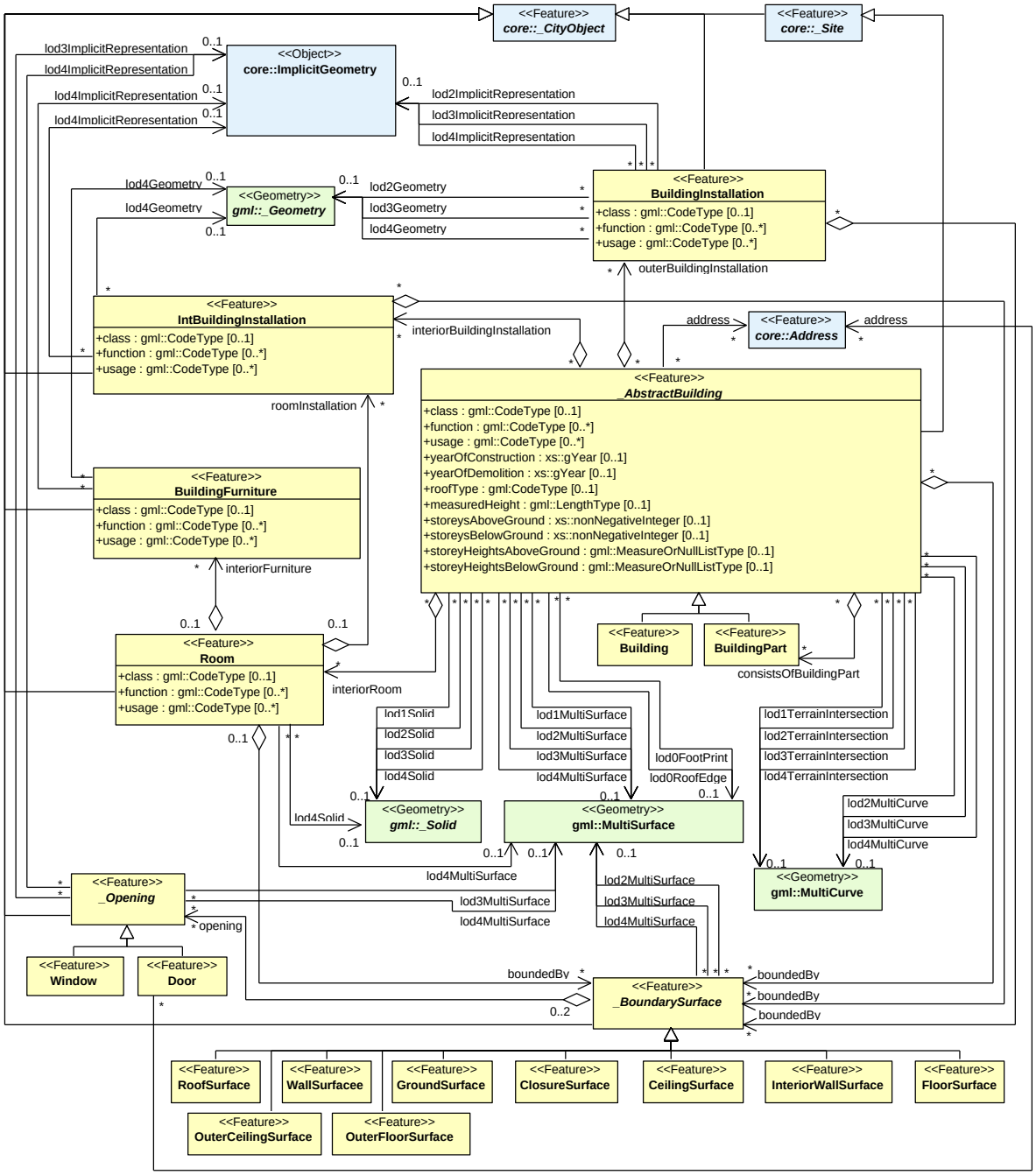

Fig. 27: UML diagram of CityGML's building model. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Building* module.

The geometric representation and semantic structure of an *_AbstractBuilding* is shown in Fig. 27. The model is successively refined from LOD0 to LOD4. Therefore, not all components of a building model are represented equally in each LOD and not all aggregation levels are allowed in each LOD. In CityGML, all object classes are associated to the LODs with respect to the proposed minimum acquisition criteria for each LOD (cf. chapter 6.2). An object can be represented simultaneously in different LODs by providing distinct geometries for the corresponding LODs.

In LOD0, the building can be represented by horizontal, 3-dimensional surfaces. These can represent the footprint of the building and, separately, the roof edge. This allows the easy integration of 2D data into the model. In many countries these 2D geometries readily exist, for example in cadastral or topographic data holdings. Cadastre data typically depicts the shape of the building on the ground (footprints) and topographic data is often a mixture between footprints and geometries at roof level (roof edges), which are often photogrametrically extracted from area/satellite images or derived from airborne laser data. The building model allows the inclusion of

both. In this case large overhanging roofs can be modelled as a preliminary stage to more detailed LOD2 and LOD3 depictions. The surface geometries require 3D coordinates, though it is mandated that the height values of all vertices belonging to the same surface are identical. If 2D geometries are imported into any of these two LOD0 geometries, an appropriate height value for all vertices needs to be chosen. The footprint is typically located at the lowest elevation of the ground surface of the building whereas the roof edge representation should be placed at roof level (e.g., eaves height).

In LOD1, a building model consists of a generalized geometric representation of the outer shell. Optionally, a *gml:MultiCurve* representing the *TerrainIntersectionCurve* (cf. chapter 6.5) can be specified. This geometric representation is refined in LOD2 by additional *gml:MultiSurface* and *gml:MultiCurve* geometries, used for modelling architectural details like roof overhangs, columns, or antennas. In LOD2 and higher LODs the outer facade of a building can also be differentiated semantically by the classes *_BoundarySurface* and *BuildingInstallation*. A *_BoundarySurface* is a part of the building's exterior shell with a special function like wall (*WallSurface*), roof (*RoofSurface*), ground plate (*GroundSurface*), outer floor (*OuterFloorSurface*), outer ceiling (*OuterCeilingSurface*) or *ClosureSurface*. The *BuildingInstallation* class is used for building elements like balconies, chimneys, dormers or outer stairs, strongly affecting the outer appearance of a building. A *BuildingInstallation* may have the attributes *class*, *function*, and *usage* (cf. Fig. 27).

In LOD3, the openings in *_BoundarySurface* objects (doors and windows) can be represented as thematic objects. In LOD4, the highest level of resolution, also the interior of a building, composed of several rooms, is represented in the building model by the class *Room*. This enlargement allows a virtual accessibility of buildings, e.g. for visitor information in a museum ("Location Based Services"), the examination of accommodation standards or the presentation of daylight illumination of a building. The aggregation of rooms according to arbitrary, user defined criteria (e.g. for defining the rooms corresponding to a certain storey) is achieved by employing the general grouping concept provided by CityGML (cf. chapter 10.3.6). Interior installations of a building, i.e. objects within a building which (in contrast to furniture) cannot be moved, are represented by the class *IntBuildingInstallation*. If an installation is attached to a specific room (e.g. radiators or lamps), they are associated with the *Room* class, otherwise (e.g. in case of rafters or pipes) with *_AbstractBuilding.* A *Room* may have the attributes *class*, *function* and *usage* whose value can be defined in code lists (chapter 10.3.8 and annex C.1). The *class* attribute allows a classification of rooms with respect to the stated function, e.g. commercial or private rooms, and occurs only once. The *function* attribute is intended to express the main purpose of the room, e.g. living room, kitchen. The attribute *usage* can be used if the way the object is actually used differs from the function. Both attributes can occur multiple times.

The visible surface of a room is represented geometrically as a *Solid* or *MultiSurface*. Semantically, the surface can be structured into specialised *_BoundarySurfaces*, representing floor (*FloorSurface*), ceiling (*CeilingSurface*), and interior walls (*InteriorWallSurface*). Room furniture, like tables and chairs, can be represented in the CityGML building model with the class *BuildingFurniture*. A *BuildingFurniture* may have the attributes *class*, *function* and *usage*. Annexes G.1 to G.6 provide example CityGML documents containing a single building model which is subsequently refined from a coarse LOD0 representation up to a semantically rich and geometric-topologically sound LOD4 model including the building interior.

XML namespace

The XML namespace of the CityGML *Building* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/building/2.0. Within the XML Schema definition of the *Building* module, this URI is also used to identify the default namespace.

10.3.1 Building and building part

BuildingType, Building

```
<xs:complexType name="BuildingType">
  <xs:complexContent>
    <xs:extension base="AbstractBuildingType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfBuilding" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
```



```
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Building" type="BuildingType" substitutionGroup="_AbstractBuilding"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBuilding" type="xs:anyType" abstract="true"/>
```

The *Building* class is one of the two subclasses of *_AbstractBuilding*. If a building only consists of one (homogeneous) part, this class shall be used. A building composed of structural segments differing in, for example the number of storeys or the roof type has to be separated into one *Building* having one or more additional *BuildingPart* (see Fig. 28). The geometry and non-spatial properties of the central part of the building should be represented in the aggregating *Building* feature.

BuildingPartType, BuildingPart

```
<xs:complexType name="BuildingPartType">
  <xs:complexContent>
    <xs:extension base="AbstractBuildingType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfBuildingPart" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="BuildingPart" type="BuildingPartType" substitutionGroup="_AbstractBuilding"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBuildingPart" type="xs:anyType" abstract="true"/>
```

The class *BuildingPart* is derived from *_AbstractBuilding*. It is used to model a structural part of a building (see Fig. 28). A *BuildingPart* object should be uniquely related to exactly one building or building part object.

Fig. 28: Examples of buildings consisting of one and two building parts (source: City of Coburg).

AbstractBuildingType, _AbstractBuilding

```
<xs:complexType name="AbstractBuildingType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractSiteType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="yearOfConstruction" type="xs:gYear" minOccurs="0"/>
        <xs:element name="yearOfDemolition" type="xs:gYear" minOccurs="0"/>
        <xs:element name="roofType" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="measuredHeight" type="gml:LengthType" minOccurs="0"/>
        <xs:element name="storeysAboveGround" type="xs:nonNegativeInteger" minOccurs="0"/>
        <xs:element name="storeysBelowGround" type="xs:nonNegativeInteger" minOccurs="0"/>
        <xs:element name="storeyHeightsAboveGround" type="gml:MeasureOrNullListType" minOccurs="0"/>
```

```
            <xs:element name="storeyHeightsBelowGround" type="gml:MeasureOrNullListType" minOccurs="0"/>
            <xs:element name="lod0FootPrint" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod0RoofEdge" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod1Solid" type="gml:SolidPropertyType" minOccurs="0"/>
            <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod2Solid" type="gml:SolidPropertyType" minOccurs="0"/>
            <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod2MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="outerBuildingInstallation" type="BuildingInstallationPropertyType" minOccurs="0"
                        maxOccurs="unbounded"/>
            <xs:element name="interiorBuildingInstallation" type="IntBuildingInstallationPropertyType" minOccurs="0"
                        maxOccurs="unbounded"/>
            <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="lod3Solid" type="gml:SolidPropertyType" minOccurs="0"/>
            <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod3MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
            <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod4MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="interiorRoom" type="InteriorRoomPropertyType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="consistsOfBuildingPart" type="BuildingPartPropertyType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="address" type="core:AddressPropertyType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="_GenericApplicationPropertyOfAbstractBuilding" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:element name="_AbstractBuilding" type="AbstractBuildingType" abstract="true" substitutionGroup="core:_Site"/>
    <!-- ========================================================================================== -->
    <xs:element name="_GenericApplicationPropertyOfAbstractBuilding" type="xs:anyType" abstract="true"/>
```

The abstract class *_AbstractBuilding* contains properties for building attributes, purely geometric representations, and geometric/semantic representations of the building or building part in different levels of detail. The attributes describe:

a) The classification of the building or building part (*class*), the different intended usages (*function*), and the different actual usages (*usage*). The permitted values for these attributes can be specified in code lists.

b) The year of construction (*yearOfConstruction*) and the year of demolition (*yearOfDemolition*) of the building or building part. These attributes can be used to describe the chronology of the building development within a city model. The points of time refer to real world time.

c) The roof type of the building or building part (*roofType*). The permitted values for this attribute can be specified in a code list.

d) The measured relative height (*measuredHeight*) of the building or building part.

e) The number of storeys above (*storeyAboveGround*) and below (*storeyBelowGround*) ground level.

f) The list of storey heights above (*storeyHeightsAboveGround*) and below (*storeyHeightsBelowGround*) ground level. The first value in a list denotes the height of the nearest storey wrt. to the ground level and last value the height of the farthest.

Spanning the different levels of detail, the building model differs in the complexity and granularity of the geometric representation and the thematic structuring of the model into components with a special semantic meaning. This is illustrated in Fig. 29 and Fig. 30, showing the same building in five different LODs. The class *_AbstractBuilding* has a number of properties which are associated with certain LODs.

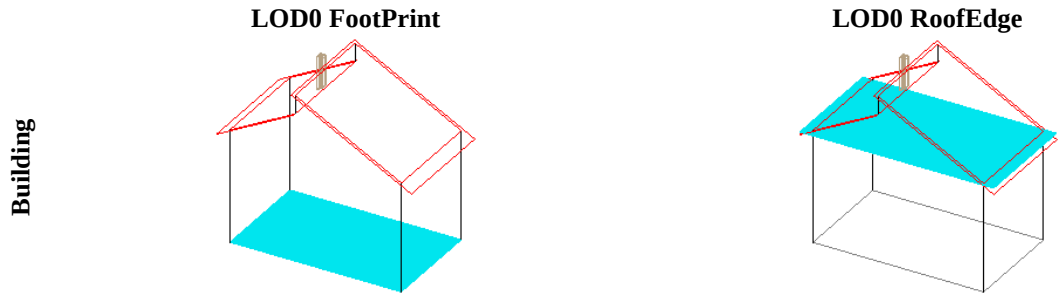

Fig. 29: The two possibilities of modeling a building in LOD0 using horizontal 3D surfaces. On the left, the building footprint (*lod0FootPrint*) is shown (cyan) which denotes the shape of the building on the ground. The corresponding surface representation is located at ground level. On the right, the *lod0RoofEgde* representation is illustrated (cyan) which results from a horizontal projection of the building's roof and which is located at the eaves height (source: Karlsruhe Institute of Technology (KIT), courtesy of Franz-Josef Kaiser).

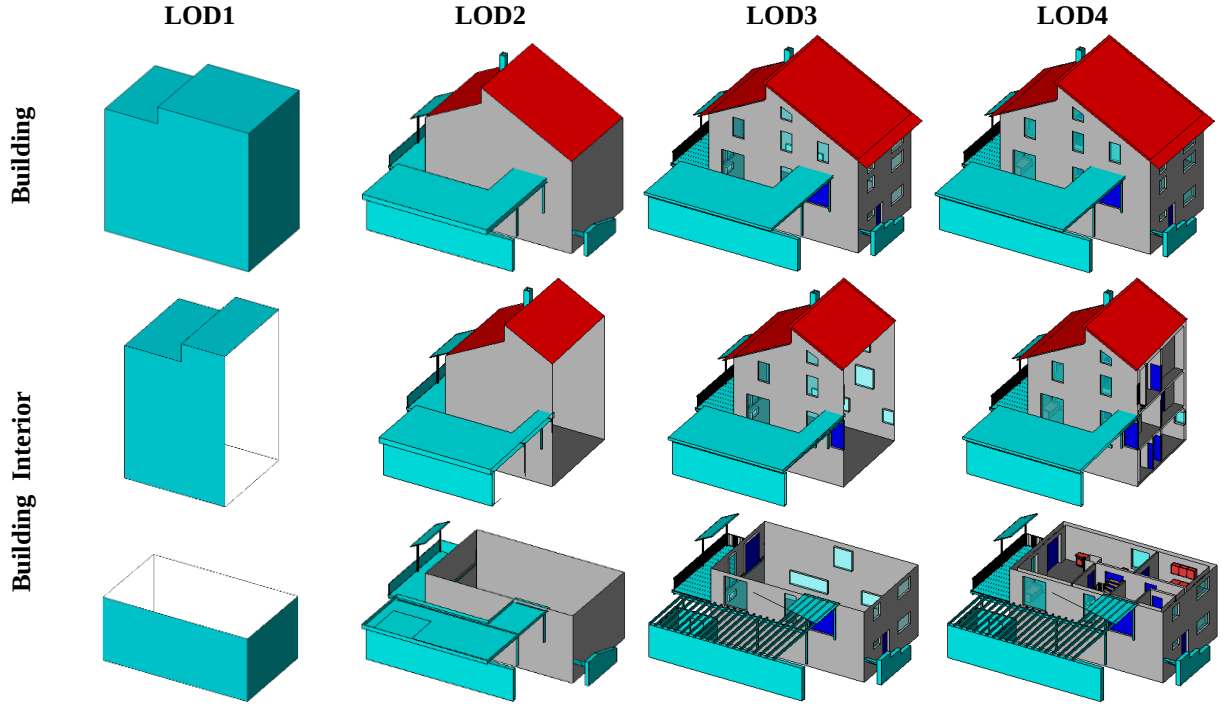

Fig. 30: Building model in LOD1 – LOD4 (source: Karlsruhe Institute of Technology (KIT), courtesy of Franz-Josef Kaiser).

Tab. 5 shows the correspondence of the different geometric and semantic themes of the building model to LODs. In LOD1 – 4, the volume of a building can be expressed by a *gml:Solid* geometry and/or a *gml:MultiSurface* geometry. The definition of a 3D Terrain Intersection Curve (TIC), used to integrate buildings from different sources with the Digital Terrain Model, is also possible in LOD1 – 4. The TIC can – but does not have to – build closed rings around the building or building parts.

In LOD0 (cf. Fig. 29) the building is represented by horizontal surfaces describing the footprint and the roof edge.

In LOD1 (cf. Fig. 30), the different structural entities of a building are aggregated to a simple block and not differentiated in detail. The volumetric and surface parts of the exterior building shell are identical and only one of the corresponding properties (*lod1Solid* or *lod1MultiSurface*) must be used.

In LOD2 and higher levels of detail, the exterior shell of a building is not only represented geometrically as *gml:Solid* geometry and/or a *gml:MultiSurface* geometry, but it can also be composed of semantic objects. The base class for all objects semantically structuring the building shell is *_BoundarySurface* (cf. chapter 10.3.2), which is associated with a *gml:MultiSurface* geometry. If in a building model there is both a geometric representation of the exterior shell as volume or surface model and a semantic representation by means of thematic *_BoundarySurfaces*, the geometric representation must not explicitly define the geometry, but has to reference the corresponding geometry components of the *gml:MultiSurface* of the *_BoundarySurface* elements.

Geometric / semantic theme	Property type	LOD0	LOD1	LOD2	LOD3	LOD4
Building footprint and roof edge	*gml:MultiSurfaceType*	•				
Volume part of the building shell	*gml:SolidType*		•	•	•	•
Surface part of the building shell	*gml:MultiSurfaceType*		•	•	•	•
Terrain intersection curve	*gml:MultiCurveType*		•	•	•	•
Curve part of the building shell	*gml:MultiCurveType*			•	•	•
Building parts	*BuildingPartType*		•	•	•	•
Boundary surfaces (chapter 10.3.3)	*AbstractBoundarySurfaceType*			•	•	•
Outer building installations (chapter 10.3.2)	*BuildingInstallationType*			•	•	•
Openings (chapter 10.3.4)	*AbstractOpeningType*				•	•
Rooms *(*chapter 10.3.5*)*	*RoomType*					•
Interior building installations (chapter 10.3.5)	*IntBuildingInstallationType*					•

Tab. 5: Semantic themes of the class *_AbstractBuilding*.

Apart from *BuildingParts*, smaller features of the building ("outer building installations") can also strongly affect the building characteristic. These features are modelled by the class *BuildingInstallation* (cf. chapter 10.3.2). Typical candidates for this class are chimneys (see. Fig. 30), dormers (see Fig. 28), balconies, outer stairs, or antennas. *BuildingInstallations* may only be included in LOD2 models, if their extents exceed the proposed minimum dimensions as specified in chapter 6.2. For the geometrical representation of the class *BuildingInstallation*, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used.

The class *_AbstractBuilding* has no additional properties for LOD3. Besides the higher requirements on geometric precision and smaller minimum dimensions, the main difference of LOD2 and LOD3 buildings concerns the class *_BoundarySurface* (cf. chapter 10.3.3). In LOD3, openings in a building corresponding with windows or doors (see Fig. 30) are modelled by the abstract class *_Opening* and the derived subclasses *Window* and *Door* (cf. chapter 10.3.4).

With respect to the exterior building shell, the LOD4 data model is identical to that of LOD3. But LOD4 provides the possibility to model the interior structure of a building with the classes *IntBuildingInstallation* and *Room* (cf. chapter 10.3.5).

Each *Building* or *BuildingPart* feature may be assigned zero or more addresses using the *address* property. The corresponding *AddressPropertyType* is defined within the CityGML core module (cf. chapter 10.1.4).

10.3.2 Outer building installations

BuildingInstallationType, BuildingInstallation

```
<xs:complexType name="BuildingInstallationType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
```

```
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="BuildingInstallation" type="BuildingInstallationType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBuildingInstallation" type="xs:anyType" abstract="true"/>
```

A *BuildingInstallation* is an outer component of a building which has not the significance of a *BuildingPart*, but which strongly affects the outer characteristic of the building. Examples are chimneys, stairs, antennas, balconies or attached roofs above stairs and paths. A *BuildingInstallation* optionally has attributes *class*, *function* and *usage*. The attribute *class* - which can only occur once - represents a general classification of the installation. With the attributes *function* and *usage*, nominal and real functions of a building installation can be described. For all three attributes the list of feasible values can be specified in a code list. For the geometrical representation of a *BuildingInstallation*, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used. Alternatively, the geometry may be given as *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype building installation is stored only once in a local coordinate system and referenced by other building installation features (see chapter 8.2). The visible surfaces of a building installation can be semantically classified using the concept of boundary surfaces (cf. 10.3.3). A *BuildingInstallation* object should be uniquely related to exactly one building or building part object.

10.3.3 Boundary surfaces

AbstractBoundarySurfaceType, _BoundarySurface

```
<xs:complexType name="AbstractBoundarySurfaceType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="opening" type="OpeningPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfBoundarySurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_BoundarySurface" type="AbstractBoundarySurfaceType" abstract="true" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBoundarySurface" type="xs:anyType" abstract="true"/>
```

_BoundarySurface is the abstract base class for several thematic classes, structuring the exterior shell of a building as well as the visible surfaces of rooms and both outer and interior building installations. It is a subclass of *_CityObject* and thus inherits all properties like the GML3 standard feature properties (*gml:name* etc.) and the CityGML specific properties like *ExternalReferences*. From *_BoundarySurface*, the thematic classes *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface*, *ClosureSurface*, *FloorSurface*, *InteriorWallSurface*, and *CeilingSurface* are derived. The thematic classification of building surfaces is illustrated in Fig. 31 (outer building shell) and Fig. 32 (additional interior surfaces) and subsequently specified.

For each LOD between 2 and 4, the geometry of a *_BoundarySurface* may be defined by a different *gml:MultiSurface* geometry.

In LOD3 and LOD4, a *_BoundarySurface* may contain *_Openings* (cf. chapter 10.3.4) like doors and windows. If the geometric location of *_Openings* topologically lies within a surface component (e.g. *gml:Polygon*) of the *gml:MultiSurface* geometry, these *_Openings* must be represented as holes within that surface. A hole is represented by an interior ring within the corresponding surface geometry object. According to GML3, the points have to be specified in reverse order (exterior boundaries counter-clockwise and interior boundaries clockwise when looking in opposite direction of the surface's normal vector). If such an opening is sealed by a *Door*, a *Window*, or a *ClosureSurface*, their outer boundary may consist of the same points as the inner ring (denoting the hole) of the surrounding surface. The embrasure surfaces of an *Opening* belong to the relevant adjacent *_BoundarySurface*. If, for example a door seals the *Opening*, the embrasure surface on the one side of the door belongs to the *InteriorWallSurface* and on the other side to the *WallSurface* (Fig. 32 on the right).

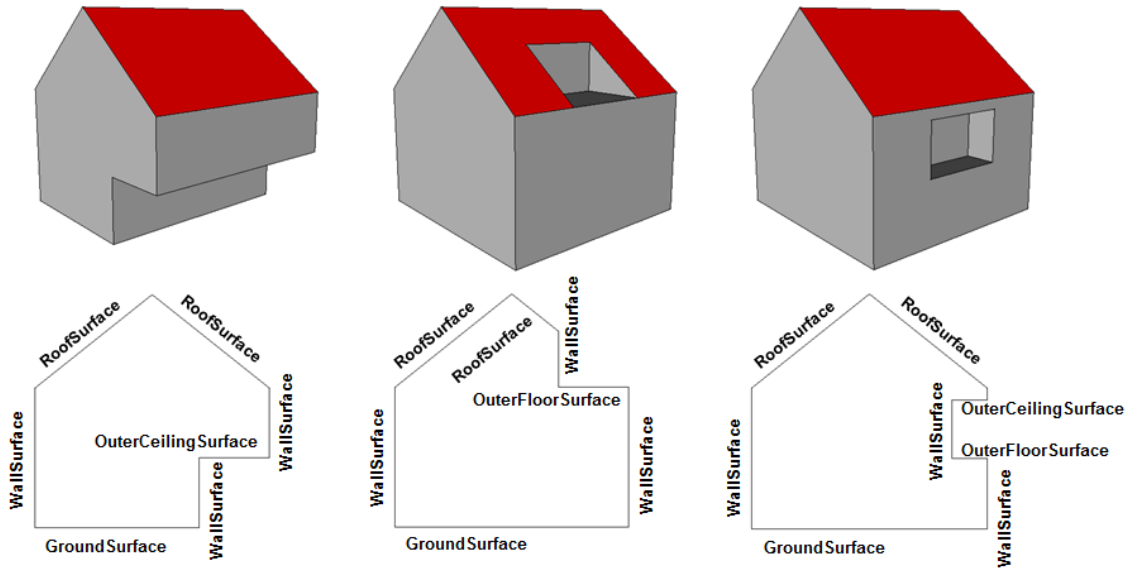

Fig. 31: Examples of the classification of *_BoundarySurfaces* of the outer building shell (source: Karlsruhe Institute of Technology (KIT))

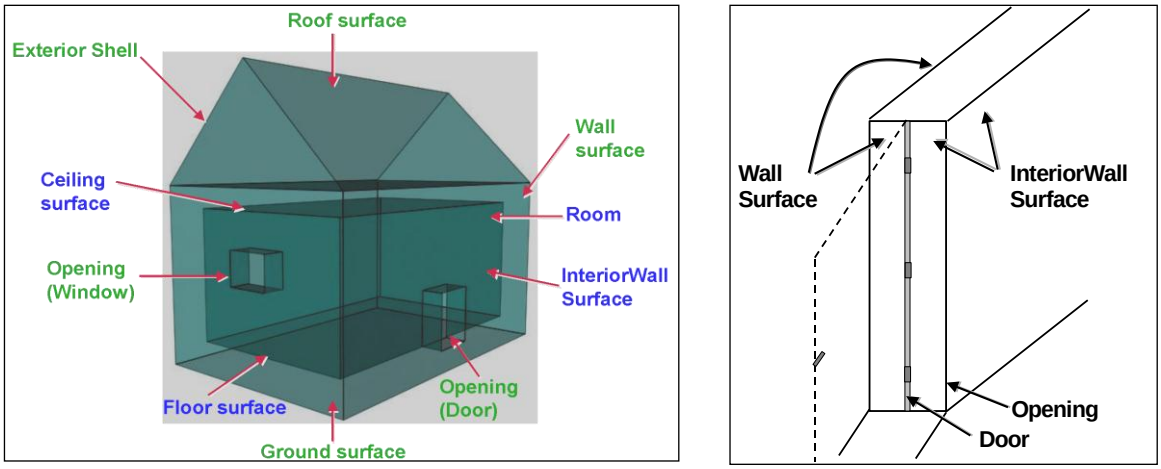

Fig. 32: Classification of *BoundarySurfaces* (left), in particular for *Openings* (right) (graphic: IGG Uni Bonn).

GroundSurfaceType, GroundSurface

```
<xs:complexType name="GroundSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="GroundSurface" type="GroundSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfGroundSurface" type="xs:anyType" abstract="true"/>
```

The ground plate of a building or building part is modelled by the class *GroundSurface*. The polygon defining the ground plate is congruent with the building's footprint. However, the surface normal of the ground plate is pointing downwards.

OuterCeilingSurfaceType, OuterCeilingSurface

```
<xs:complexType name="OuterCeilingSurfaceType">
  <xs:complexContent>
```

```
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfOuterCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="OuterCeilingSurface" type="OuterCeilingSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfOuterCeilingSurface" type="xs:anyType" abstract="true"/>
```

A mostly horizontal surface belonging to the outer building shell and having the orientation pointing downwards can be modeled as an *OuterCeilingSurface*. Examples are the visible part of the ceiling of a loggia or the ceiling of a passage.

WallSurfaceType, WallSurface

```
<xs:complexType name="WallSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWallSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="WallSurface" type="WallSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfWallSurface" type="xs:anyType" abstract="true"/>
```

All parts of the building facade belonging to the outer building shell can be modelled by the class *WallSurface*.

OuterFloorSurfaceType, OuterFloorSurface

```
<xs:complexType name="OuterFloorSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfOuterFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="OuterFloorSurface" type="OuterFloorSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfOuterFloorSurface" type="xs:anyType" abstract="true"/>
```

A mostly horizontal surface belonging to the outer building shell and with the orientation pointing upwards can be modeled as an *OuterFloorSurface*. An example is the floor of a loggia.

RoofSurfaceType, RoofSurface

```
<xs:complexType name="RoofSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfRoofSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="RoofSurface" type="RoofSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfRoofSurface" type="xs:anyType" abstract="true"/>
```

The major roof parts of a building or building part are expressed by the class *RoofSurface*. Secondary parts of a roof with a specific semantic meaning like dormers or chimneys should be modelled as *BuildingInstallation*.

ClosureSurfaceType, ClosureSurface

```
<xs:complexType name="ClosureSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="ClosureSurface" type="ClosureSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfClosureSurface" type="xs:anyType" abstract="true"/>
```

An opening in a building not filled by a door or window can be sealed by a virtual surface called *ClosureSurface* (cf. chapter 6.4). Hence, buildings with open sides like a barn or a hangar, can be virtually closed in order to be able to compute their volume. *ClosureSurfaces* are also used in the interior building model. If two rooms with a different function (e.g. kitchen and living room) are directly connected without a separating door, a *ClosureSurface* should be used to separate or connect the volumes of both rooms.

FloorSurfaceType, FloorSurface

```
<xs:complexType name="FloorSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="FloorSurface" type="FloorSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfFloorSurface" type="xs:anyType" abstract="true"/>
```

The class *FloorSurface* must only be used in the LOD4 interior building model for modelling the floor of a room.

InteriorWallSurfaceType, InteriorWallSurface

```
<xs:complexType name="InteriorWallSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfInteriorWallSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="InteriorWallSurface" type="InteriorWallSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfInteriorWallSurface" type="xs:anyType" abstract="true"/>
```

The class *InteriorWallSurface* must only be used in the LOD4 interior building model for modelling the visible surfaces of the room walls.

CeilingSurfaceType, CeilingSurface

```
<xs:complexType name="CeilingSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="CeilingSurface" type="CeilingSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfCeilingSurface" type="xs:anyType" abstract="true"/>
```

The class *CeilingSurface* must only be used in the LOD4 interior building model for modelling the ceiling of a room.

10.3.4 Openings

AbstractOpeningType, _Opening

```
<xs:complexType name="AbstractOpeningType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfOpening" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_Opening" type="AbstractOpeningType" abstract="true" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfOpening" type="xs:anyType" abstract="true"/>
```

The class *_Opening* is the abstract base class for semantically describing openings like doors or windows in outer or inner boundary surfaces like walls and roofs. Openings only exist in models of LOD3 or LOD4. Each *_Opening* is associated with a *gml:MultiSurface* geometry. Alternatively, the geometry may be given as *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype opening is stored only once in a local coordinate system and referenced by other opening features (see chapter 8.2).

WindowType, Window

```
<xs:complexType name="WindowType">
  <xs:complexContent>
    <xs:extension base="AbstractOpeningType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWindow" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Window" type="WindowType" substitutionGroup="_Opening"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfWindow" type="xs:anyType" abstract="true"/>
```

The class *Window* is used for modelling windows in the exterior shell of a building, or hatches between adjacent rooms. The formal difference between the classes *Window* and *Door* is that – in normal cases – *Windows* are not specifically intended for the transit of people or vehicles.

DoorType, Door

```
<xs:complexType name="DoorType">
  <xs:complexContent>
    <xs:extension base="AbstractOpeningType">
      <xs:sequence>
        <xs:element name="address" type="core:AddressPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfDoor" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Door" type="DoorType" substitutionGroup="_Opening"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfDoor" type="xs:anyType" abstract="true"/>
```

The class *Door* is used for modelling doors in the exterior shell of a building, or between adjacent rooms. Doors can be used by people to enter or leave a building or room. In contrast to a *ClosureSurface* a door may be closed, blocking the transit of people. A *Door* may be assigned zero or more addresses. The corresponding *Address-PropertyType* is defined within the CityGML core module (cf. chapter 10.1.4) .

10.3.5 Building interior

RoomType, Room

```
<xs:complexType name="RoomType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="interiorFurniture" type="InteriorFurniturePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="roomInstallation" type="IntBuildingInstallationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfRoom" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Room" type="RoomType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfRoom" type="xs:anyType" abstract="true"/>
```

A *Room* is a semantic object for modelling the free space inside a building and should be uniquely related to exactly one building or building part object. It should be closed (if necessary by using *ClosureSurfaces*) and the geometry normally will be described by a solid (*lod4Solid*). However, if the topological correctness of the boundary cannot be guaranteed, the geometry can alternatively be given as a *MultiSurface* (*lod4MultiSurface*). The surface normals of the outer shell of a GML solid must point outwards. This is important to consider when *Room* surfaces should be assigned *Appearances*. In this case, textures and colors must be placed on the backside of the corresponding surfaces in order to be visible from the inside of the room.

In addition to the geometrical representation, different parts of the visible surface of a room can be modelled by specialised *BoundarySurfaces* (*FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface* cf. chapter 10.3.3).

A special task is the modelling of passages between adjacent rooms. The room solids are topologically connected by the surfaces representing hatches, doors or closure surfaces that seal open doorways. Rooms are defined as being adjacent, if they have common *_Openings* or *ClosureSurfaces*. The surface that represents the opening geometrically is part of the boundaries of the solids of both rooms, or the opening is referenced by both rooms on the semantic level. This adjacency implies an accessibility graph, which can be employed to determine the

spread of e.g. smoke or gas, but which can also be used to compute escape routes using classical shortest path algorithms (see Fig. 33).

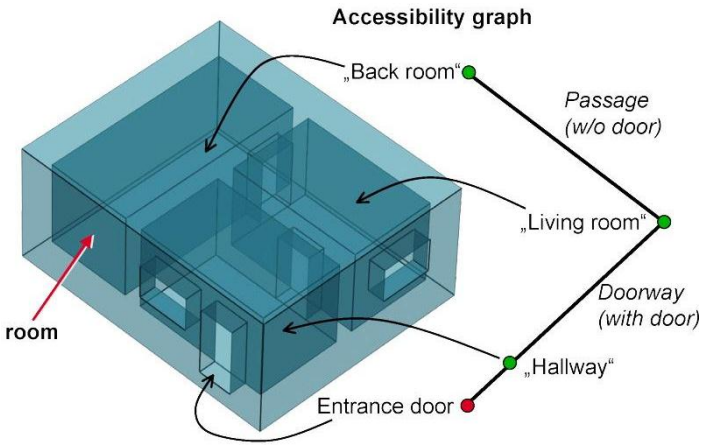

Fig. 33: Accessibility graph derived from topological adjacencies of room surfaces (graphic: IGG Uni Bonn).

BuildingFurnitureType, BuildingFurniture

```
<xs:complexType name="BuildingFurnitureType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfBuildingFurniture" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="BuildingFurniture" type="BuildingFurnitureType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBuildingFurniture" type="xs:anyType" abstract="true"/>
```

Rooms may have *BuildingFurnitures* and *IntBuildingInstallations*. A *BuildingFurniture* is a movable part of a room, such as a chair or furniture. A *BuildingFurniture* object should be uniquely related to exactly one room object. Its geometry may be represented by an explicit geometry or an *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype building furniture is stored only once in a local coordinate system and referenced by other building furniture features (see chapter 8.2).

IntBuildingInstallationType, IntBuildingInstallation

```
<xs:complexType name="IntBuildingInstallationType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfIntBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="IntBuildingInstallation" type="IntBuildingInstallationType" substitutionGroup="core:_CityObject"/>
```



```
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfIntBuildingInstallation" type="xs:anyType" abstract="true"/>
```

An *IntBuildingInstallation* is an object inside a building with a specialised function or semantic meaning. In contrast to *BuildingFurniture*, *IntBuildingInstallations* are permanently attached to the building structure and cannot be moved. Typical examples are interior stairs, railings, radiators or pipes. Objects of the class *IntBuildingInstallation* can either be associated with a room (class *Room*), or with the complete building / building part (class *_AbstractBuilding*, cf. chapter 10.3.1). However, they should be uniquely related to exactly one room or one building / building part object. An *IntBuildingInstallation* optionally has attributes *class*, *function* and *usage*. The attribute *class*, which can only occur once, represents a general classification of the internal building component. With the attributes *function* and *usage*, nominal and real functions of a building installation can be described. For all three attributes the list of feasible values can be specified in a code list. For the geometrical representation of an *IntBuildingInstallation*, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used. Alternatively, the geometry may be given as *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype interior building installation is stored only once in a local coordinate system and referenced by other interior building installation features (see chapter 8.2). The visible surfaces of an interior building installation can be semantically classified using the concept of boundary surfaces (cf. 10.3.3).

10.3.6 Modelling building storeys using CityObjectGroups

CityGML does currently not provide a specific concept for the representation of storeys as it is available in the AEC/FM standard IFC (IAI 2006). However, a storey can be represented as an explicit aggregation of all building features on a certain height level using CityGML's notion of *CityObjectGroups* (cf. chapter 10.11). This would include *Rooms*, *Doors*, *Windows*, *IntBuildingInstallations* and *BuildingFurniture*. If thematic surfaces like walls and interior walls should also be associated to a specific storey, this might require the vertical fragmentation of these surfaces (one per storey), as in virtual 3D city models they typically span the whole façade.

In order to model building storeys with CityGML's generic grouping concept, a nested hierarchy of *CityObjectGroup* objects has to be used. In a first step, all semantic objects belonging to a specific storey are grouped. The attributes of the corresponding *CityObjectGroup* object are set as follows:

- The *class* attribute shall be assigned the value "*building separation*".
- The *function* attribute shall be assigned the value "*lodXStorey*" with X between 1 and 4 in order to denote that this group represents a storey wrt. a specific LOD.
- The storey name or number can be stored in the *gml:name* property. The storey number attribute shall be assigned the value *"storeyNo_X"* with decimal number X in order to denote that this group represents a storey wrt. a specific number.

In a second step, the *CityObjectGroup* objects representing different storeys are grouped themselves. By using the generic aggregation concept of *CityObjectGroup*, the "storeys group" is associated with the corresponding *Building* or *BuildingPart* object. The *class* attribute of the storeys group shall be assigned the value "*building storeys*".

10.3.7 Examples

The LOD1 model of the Campus North of the Karlruhe Institute of Technology (KIT) shown in Fig. 34 consists of 596 buildings and 187 building parts. The footprint geometries of the buildings are taken from a cadastral information system and extruded by a given height. Buildings with a unique identifier and a single height value are modeled as one building (*bldg:Building*). Buildings having a unique identifier but different height values are modeled as one building (*bldg:Building*) with one or more building parts (*bldg:BuildingPart*). Both buildings and building parts have solid geometries and their height values are additionally represented as thematic attribute (*bldg:measuredHeight*). Fig. 34 shows an aerial photograph of the KIT Campus North (left) and the CityGML LOD1 model (right).

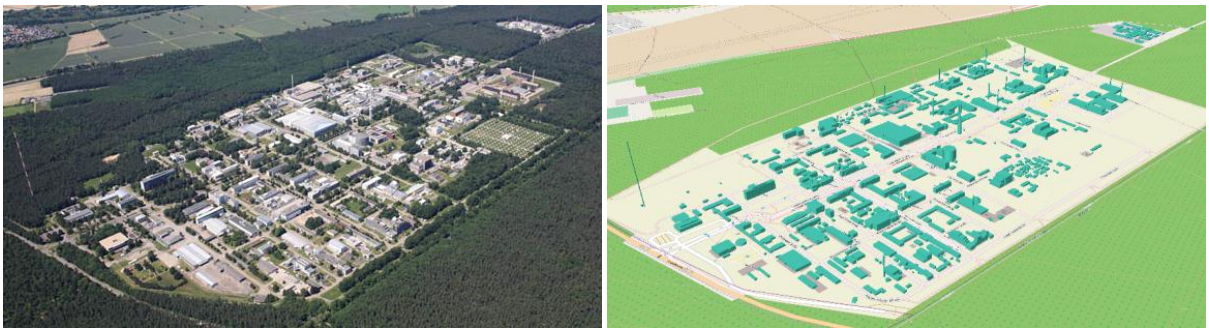

Fig. 34: LOD1 model of the KIT Campus North (source: Karlsruhe Institute of Technology (KIT)).

An example for a fully textured LOD2 building model is given in Fig. 35 which shows the Bernhardus church located in the city of Karlsruhe, Germany. On the left side of Fig. 35, a photograph of the church in real world is shown whereas the CityGML building model of the church with photorealistic textures is illustrated on the right. The model is bounded by a ground surface, several wall and roof surfaces. The railing above the church clock is modeled as a building installation (*BuildingInstallation*).

Fig. 35: Textured LOD2 model of the Bernhardus church in Karlsruhe (source: Karlsruhe Institute of Technology (KIT), courtesy of City of Karlsruhe).

The model shown in Fig. 36 was derived from a 3D CAD model generated during the planning phase of the building. On the left side of Fig. 36, the building is shown whereas on the right side the LOD3 model is presented. The building itself is bounded by wall surfaces, roof surfaces and a ground surface. Doors and windows are modeled including reveals. According to the cadaster data, the car port next to the building is not part of the building. Therefore the car port, the balcony and the chimney are modeled as building installations (*BuildingInstallation*). The model also contains the terrain intersection curve (*lod3TerrainIntersection*) as planned by the architect.

In order to determine the volume of the building, the geometries of all boundary surfaces, including doors and windows, are referenced by the building solid (*lod3Solid*) using the XLink mechanism. Consequently, the roof surfaces are split into surfaces representing the roof itself and surfaces representing the roof overhangs.

Fig. 36: Example of a building modeled in the Level of Detail 3. The chimney, the balcony and the car port are modeled as building installations (source: Karlsruhe Institute of Technology (KIT), courtesy of Franz-Josef Kaiser).

10.3.8 Code lists

The attributes *class*, *function*, *usage*, and *roofType* of the feature *_AbstractBuilding* as well as the attributes *class*, *function* and *usage* of the features *BuildingInstallation*, *Room*, *BuildingFurniture* and *IntBuildingInstallation* are specified as *gml:CodeType*. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.1.

10.3.9 Conformance requirements

Base requirements

1. If a building only consists of one (homogeneous) part, it shall be represented by the element *Building*. However, if a building is composed of individual structural segments, it shall be modelled as a *Building* element having one or more additional *BuildingPart* elements. Only the geometry and non-spatial properties of the main part of the building should be represented within the aggregating *Building* element.

Usage restriction of building model components according to different LODs

2. The *gml:MultiSurface* geometries that are associated using the *lod0FootPrint* and *lod0RoofEdge* properties must have 3D coordinates. For each surface, the height values of the coordinate tuples belonging to the same surface shall be identical.
3. The *lodXSolid* and *lodXMultiSurface*, X $\in$ [1..4], properties (*gml:SolidPropertyType* resp. *gml:MultiSurfacePropertyType*) of *_AbstractBuilding* may be used to geometricllay represent the exterior shell of a building (as volume or surface model) within each LOD. For LOD1, either *lod1Solid* or *lod1MultiSurface* must be used, but not both. Starting from LOD2, both properties may be modelled individually and complementary.
4. Starting from LOD2, the exterior shell of an *_AbstractBuilding* may be semantically decomposed into *_BoundarySurface* elements using the *boundedBy* property (type: *BoundarySurfacePropertyType*) of *_AbstractBuilding*. Only *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface* and *ClosureSurface* as subclasses of *_BoundarySurface* are allowed. The *boundedBy* property (not to be confused with the *gml:boundedBy* property) shall not be used if the building is only represented in LOD1.

   If the exterior shell is represented by *_BoundarySurface* elements, an additional geometric representation as volume or surface model using the *lodXSolid* and *lodXMultiSurface*, X $\in$ [2..4], properties shall not explicitly define the geometry, but has to reference the according components of the *gml:MultiSurface* element of *_BoundarySurface* within each LOD using the XLink concept of GML 3.1.1.
5. Starting from LOD2, curve parts of the building shell may be represented using the *lodXMultiCurve*, X $\in$ [2..4], property of *_AbstractBuilding*. This property shall not be used if the building is only represented in LOD1.
6. Starting from LOD2, the *outerBuildingInstallation* property (type: *BuildingInstallationPropertyType*) of *_AbstractBuilding* may be used to model *BuildingInstallation* elements. *BuildingInstallation* elements shall only be used to represent outer characteristics of a building which do not have the significance of

building parts. The *outerBuildingInstallation* property shall not be used if the building is only represented in LOD1.

7. Starting from LOD2, the geometry of *BuildingInstallation* elements may be semantically classified by *_BoundarySurface* elements using the *boundedBy* property (type: *BoundarySurfacePropertyType*) of *BuildingInstallation*. Only *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface* and *ClosureSurface* as subclasses of *_BoundarySurface* are allowed.
8. Starting from LOD3, openings of *_BoundarySurface* elements may be modelled using the *opening* property (type: *OpeningPropertyType*) of *_BoundarySurface*. This property shall not be used for *_BoundarySurface* elements only represented in LOD2. Accordingly, the surface geometry representing a *_BoundarySurface* in LOD2 must be simply connected.

   The *opening* property of *_BoundarySurface* may contain or reference *_Opening* elements. If the geometric location of an *_Opening* element topologically lies within a surface component of the *_BoundarySurface*, the opening must also be represented as inner hole of that surface. The embrasure surface of an *_Opening* element shall belong to the relevant adjacent *_BoundarySurface*.
9. Starting from LOD4, the *interiorRoom* property (type: *InteriorRoomPropertyType*) of *_AbstractBuilding* may be used to semantically model the free space inside the building by *Room* elements. This property shall not be used if the building is only represented in LOD 1 – 3. The *Room* element may be geometrically represented as a surface or volume model, using its *lod4Solid* or *lod4MultiSurface* property (*gml:SolidPropertyType* resp. *gml:MultiSurfacePropertyType*).

   In addition, different parts of the visible surface of a room may be modelled by thematic *_BoundarySurface* elements. Only *FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface* as subclasses of *_BoundarySurface* are allowed. If the visible surface of a room is represented by *_BoundarySurface* elements, an additional geometric representation as volume or surface model using the *lod4Solid* and *lod4MultiSurface* property shall not explicitly define the geometry, but has to reference the according components of the *gml:MultiSurface* element of *_BoundarySurface* using the XLink concept of GML 3.1.1.
10. Starting from LOD4, the *interiorBuildingInstallation* property (type: *IntBuildingInstallationPropertyType*) of *_AbstractBuilding* may be used to represent immovable objects inside the building that are permamently attached to the building structure. The *interiorBuildingInstallation* property shall not be used if the building is only represented in LOD 1 – 3. Furthermore, the *interiorBuildingInstallation* property shall only be used if the object cannot be associated with a *Room* element. In the latter case, the *roomInstallation* property (type: *IntBuildingInstallationPropertyType*) of the corresponding *Room* element shall be used to represent the object.
11. Starting from LOD4, the geometry of *IntBuildingInstallation* elements may be semantically classified by *_BoundarySurface* elements using the *boundedBy* property (type: *BoundarySurfacePropertyType*) of *IntBuildingInstallation*. Only *FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface* as subclasses of *_BoundarySurface* are allowed.

Referential integrity

12. The *boundedBy* property (type: *BoundarySurfacePropertyType*) of the element *_AbstractBuilding* may contain a *_BoundarySurface* element inline or an XLink reference to a remote *_BoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_BoundarySurface* element (where remote *_BoundarySurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

    Only *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface* and *ClosureSurface* elements are allowed to be encapsulated or referenced by the *boundedBy* property of *_AbstractBuilding*.
13. The *outerBuildingInstallation* property (type: *BuildingInstallationPropertyType*) of the element *_AbstractBuilding* may contain a *BuildingInstallation* element inline or an XLink reference to a remote *BuildingInstallation* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *outerBuildingInstallation* property may only point to a remote *BuildingInstallation* element (where remote *BuildingInstallation* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

14. The *interiorBuildingInstallation* property (type: *IntBuildingInstallationPropertyType*) of the element *_AbstractBuilding* may contain an *IntBuildingInstallation* element inline or an XLink reference to a remote *IntBuildingInstallation* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *interiorBuildingInstallation* property may only point to a remote *IntBuildingInstallation* element (where remote *IntBuildingInstallation* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
15. The *interiorRoom* property (type: *InteriorRoomPropertyType*) of the element *_AbstractBuilding* may contain a *Room* element inline or an XLink reference to a remote *Room* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *interiorRoom* property may only point to a remote *Room* element (where remote *Room* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
16. The *consistsOfBuildingPart* property (type: *BuildingPartPropertyType*) of the element *_AbstractBuilding* may contain a *BuildingPart* element inline or an XLink reference to a remote *BuildingPart* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *consistsOfBuildingPart* property may only point to a remote *BuildingPart* element (where remote *BuildingPart* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
17. The *address* property (type: *core:AddressPropertyType*) of the element *_AbstractBuilding* may contain an *core:Address* element inline or an XLink reference to a remote *core:Address* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *address* property may only point to a remote *core:Address* element (where remote *core:Address* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
18. The *opening* property (type: *OpeningPropertyType*) of the element *_BoundarySurface* may contain an *_Opening* element inline or an XLink reference to a remote *_Opening* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *opening* property may only point to a remote *_Opening* element (where remote *_Opening* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
19. The *address* property (type: *core:AddressPropertyType*) of the element *Door* may contain an *core:Address* element inline or an XLink reference to a remote *core:Address* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *address* property may only point to a remote *core:Address* element (where remote *core:Address* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
20. The *boundedBy* property (type: *BoundarySurfacePropertyType*) of the element *BuildingInstallation* may contain a *_BoundarySurface* element inline or an XLink reference to a remote *_BoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_BoundarySurface* element (where remote *_BoundarySurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

    Only *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface* and *ClosureSurface* elements are allowed to be encapsulated or referenced by the *boundedBy* property of *BuildingInstallation*.
21. The *boundedBy* property (type: *BoundarySurfacePropertyType*) of the element *IntBuildingInstallation* may contain a *_BoundarySurface* element inline or an XLink reference to a remote *_BoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_BoundarySurface* element (where remote *_BoundarySurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

Only *FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface* elements are allowed to be encapsulated or referenced by the *boundedBy* property of *IntBuildingInstallation*.

22. The *boundedBy* property (type: *BoundarySurfacePropertyType*) of the element *Room* may contain a *_BoundarySurface* element inline or an XLink reference to a remote *_BoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_BoundarySurface* element (where remote *_BoundarySurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

    Only *FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface* elements are allowed to be encapsulated or referenced by the *boundedBy* property of *Room*.

23. The *interiorFurniture* property (type: *InteriorFurniturePropertyType*) of the element *Room* may contain an *BuildingFurniture* element inline or an XLink reference to a remote *BuildingFurniture* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *interiorFurniture* property may only point to a remote *BuildingFurniture* element (where remote *BuildingFurniture* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

24. The *roomInstallation* property (type: *IntBuildingInstallationPropertyType*) of the element *Room* may contain an *IntBuildingInstallation* element inline or an XLink reference to a remote *IntBuildingInstallation* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *roomInstallation* property may only point to a remote *IntBuildingInstallation* element (where remote *IntBuildingInstallation* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

25. The *lodXImplicitRepresentation*, X ∈ [2..4], property (type: *core:ImplicitRepresentationPropertyType*) of the element *BuildingInstallation* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lodXImplicitRepresentation*, X ∈ [2..4], property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

26. The *lod4ImplicitRepresentation* property (type: *core:ImplicitRepresentationPropertyType*) of the element *IntBuildingInstallation* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lod4ImplicitRepresentation* property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

27. The *lodXImplicitRepresentation*, X ∈ [3..4], property (type: *core:ImplicitRepresentationPropertyType*) of the element *_Opening* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lodXImplicitRepresentation*, X ∈ [3..4], property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

28. The *lod4ImplicitRepresentation* property (type: *core:ImplicitRepresentationPropertyType*) of the element *BuildingFurniture* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lod4ImplicitRepresentation* property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

10.4 Tunnel model

The tunnel model is closely related to the building model. It supports the representation of thematic and spatial aspects of tunnels and tunnel parts in four levels of detail, LOD1 to LOD4. The tunnel model of CityGML is defined by the thematic extension module *Tunnel* (cf. chapter 7). Fig. 37 provides examples of tunnel models for each LOD.

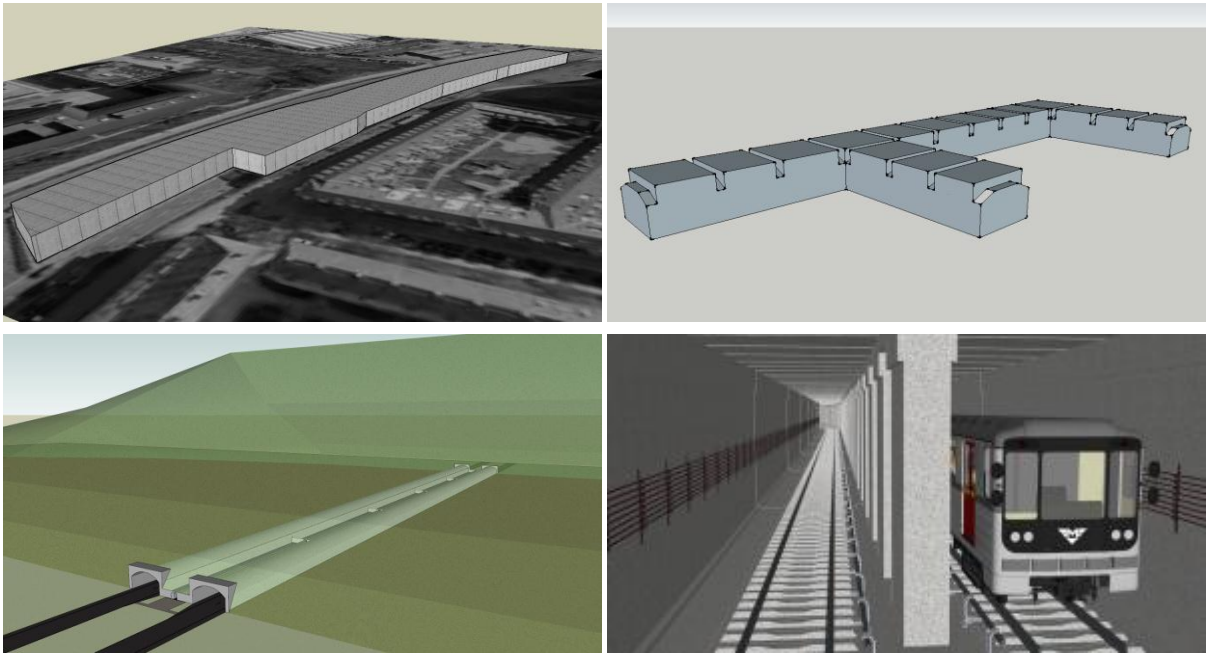

Fig. 37: Examples for tunnel models in LOD1 (upper left), LOD2 (upper right), LOD3 (lower left) and LOD4 (lower right) (source: Google 3D warehouse).

The UML diagram of the tunnel model is shown in Fig. 38. The XML schema definition is attached in annex A.11. The pivotal class of the model is *_AbstractTunnel*, which is a subclass of the thematic class *_Site* (and transitively of the root class *_CityObject*). *_AbstractTunnel* is specialized either to a *Tunnel* or to a *TunnelPart*. Since an *_AbstractTunnel* consists of *TunnelParts*, which again are *_AbstractTunnels*, an aggregation hierarchy of arbitrary depth may be realized. As subclass of the root class *_CityObject*, an *_AbstractTunnel* inherits all properties from *_CityObject* like the GML3 standard feature properties (*gml:name* etc.) and the CityGML specific properties like *ExternalReferences* (cf. chapter 6.7). Further properties not explicitly covered by *_AbstractTunnel* may be modelled as *generic attributes* provided by the CityGML *Generics* module (cf. chapter 10.12) or using the CityGML Application Domain Extension mechanism (cf. chapter 10.13).

Both classes *Tunnel* and *TunnelPart* inherit the attributes of *_AbstractTunnel*: the class of the tunnel, the function, the usage, the year of construction and the year of demolition. In contrast to *_AbstractBuilding*, *Address* features cannot be assigned to *_AbstractTunnel*.

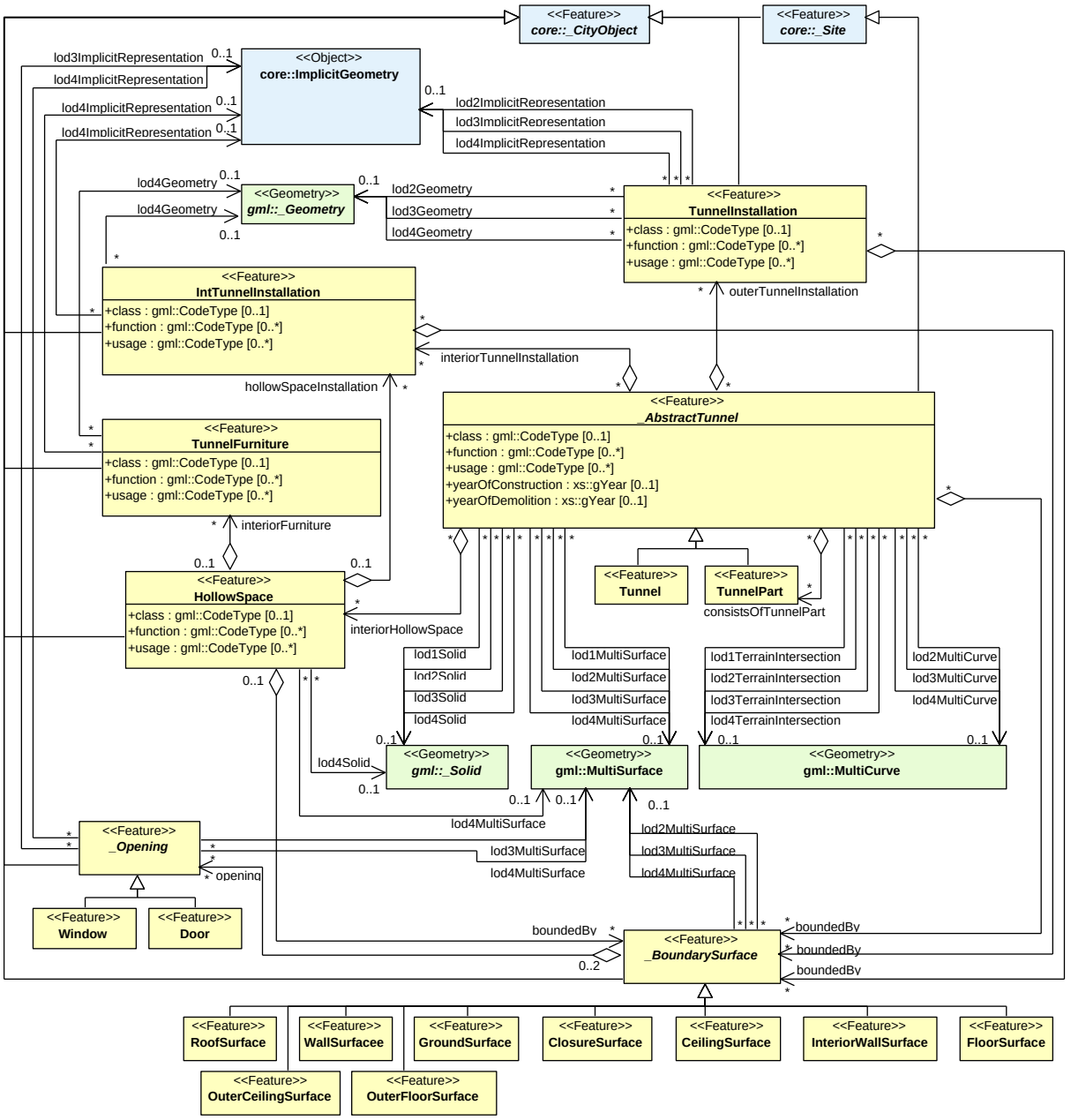

Fig. 38: UML diagram of CityGML's tunnel model. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Tunnel module.

The geometric representation and semantic structure of an *_AbstractTunnel* is shown in Fig. 38. The model is successively refined from LOD1 to LOD4. Therefore, not all components of a tunnel model are represented equally in each LOD and not all aggregation levels are allowed in each LOD. In CityGML, all object classes are associated to the LODs with respect to the proposed minimum acquisition criteria for each LOD (cf. chapter 6.2). An object can be represented simultaneously in different LODs by providing distinct geometries for the corresponding LODs.

Similar to the building and brigde models (cf. chapters 10.3 and 10.5), only the outer shell of a tunnel is represented in LOD1 – 3, which is composed of the tunnel's boundary surfaces to the surrounding earth, water, or outdoor air. The interior of a tunnel may only be modeled in LOD4. Although the interior built environment is especially relevant for subsurface objects like tunnels or underground buildings, CityGML employs a consistent LOD concept for all thematic modules. If, in contrast, the representation of the interior of subsurface objects would be possible in all LODs, the LOD concept for subsurface objects would have to substantially differ from the LOD concept for aboveground objects. This would require the precise definition of a "transition surface" which delimits the scope of both LOD concepts. Furthermore, features being partially above and below ground would have to be split into an above-ground part (modeled according to the aboveground LOD concept) and a

subsurface part (modeled according to the subsurface LOD concept). However, such a splitting violates the CityGML concept of unity of features and would not be feasible in many cases where the transition between above and below ground is often not precisely known or depends on (the LOD of) the terrain model. Hence, CityGML applies a single and consistent LOD concept to both aboveground and subsurface objects. As a consequence, penetrations between a tunnel and objects inside this tunnel (e.g., roads and railways) may occur in LOD1 – 3.

In LOD1, a tunnel model consists of a geometric representation of the tunnel volume. Optionally, a *MultiCurve* representing the *TerrainIntersectionCurve* (cf. chapter 6.5) can be specified. The geometric representation is refined in LOD2 by additional *MultiSurface* and *MultiCurve* geometries.

In LOD2 and higher LODs the outer structure of a tunnel can also be differentiated semantically by the classes *_BoundarySurface* and *TunnelInstallation*. A boundary surface is a part of the tunnel's exterior shell with a special function like wall (*WallSurface*), roof (*RoofSurface*), ground plate (*GroundSurface*), outer floor (*OuterFloorSurface*), outer ceiling (*OuterCeilingSurface*) or *ClosureSurface*. The *TunnelInstallation* class is used for tunnel elements like outer stairs, strongly affecting the outer appearance of a tunnel. A *TunnelInstallation* may have the attributes *class*, *function* and *usage* (see Fig. 38).

In LOD3, the openings in *_BoundarySurface* objects (doors and windows) can be represented as thematic objects.

In LOD4, the highest level of resolution, also the interior of a tunnel, composed of several hollow spaces, is represented in the tunnel model by the class *HollowSpace*. This enlargement allows a virtual accessibility of tunnels, e.g. for driving through a tunnel, for simulating disaster management or for presenting the light illumination within a tunnel. The aggregation of hollow spaces according to arbitrary, user defined criteria (e.g. for defining the hollow spaces corresponding to horizontal or vertical sections) is achieved by employing the general grouping concept provided by CityGML (cf. chapter 10.11). Interior installations of a tunnel, i.e. objects within a tunnel which (in contrast to furniture) cannot be moved, are represented by the class *IntTunnelInstallation*. If an installation is attached to a specific hollow space (e.g. lamps, ventilator), they are associated with the *HollowSpace* class, otherwise (e.g. pipes) with *_AbstractTunnel*. A *HollowSpace* may have the attributes *class*, *function* and *usage* whose possible values can be enumerated in code lists (chapter 10.4.7, Annex C). The *class* attribute allows a general classification of hollow spaces, e.g. commercial or private rooms, and occurs only once. The *function* attribute is intended to express the main purpose of the hollow space, e.g. control area, installation space, storage space. The attribute *usage* can be used if the way the object is actually used differs from the function. Both attributes can occur multiple times.

The visible surface of a hollow space is represented geometrically as a *Solid* or *MultiSurface*. Semantically, the surface can be structured into specialised *_BoundarySurfaces*, representing floor (*FloorSurface*), ceiling (*CeilingSurface*), and interior walls (*InteriorWallSurface*). Hollow space furniture, like movable equipment in control areas, can be represented in the CityGML tunnel model with the class *TunnelFurniture*. A *TunnelFurniture* may have the attributes *class*, *function* and *usage*.

XML namespace

The XML namespace of the CityGML Tunnel module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/tunnel/2.0. Within the XML Schema definition of the *Tunnel* module, this URI is also used to identify the default namespace.

10.4.1 Tunnel and tunnel part

TunnelType, Tunnel

```
<xs:complexType name="TunnelType">
  <xs:complexContent>
    <xs:extension base="AbstractTunnelType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfTunnel" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Tunnel" type="TunnelType" substitutionGroup="_AbstractTunnel"/>
```



```
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTunnel" type="xs:anyType" abstract="true"/>
```

The *Tunnel* class is one of the two subclasses of *_AbstractTunnel*. If a tunnel only consists of one (homogeneous) part, this class shall be used. A tunnel composed of structural segments, for example tunnel entrance and subway, has to be separated into one tunnel having one or more additional *TunnelPart* (see Fig. 39). The geometry and non-spatial properties of the central part of the tunnel should be represented in the aggregating *Tunnel* feature.

TunnelPartType, TunnelPart

```
<xs:complexType name="TunnelPartType">
  <xs:complexContent>
    <xs:extension base="AbstractTunnelType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfTunnelPart" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="TunnelPart" type="TunnelPartType" substitutionGroup="_AbstractTunnel"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTunnelPart" type="xs:anyType" abstract="true"/>
```

If sections of a tunnel differ in geometry and / or attributes, the tunnel can be separated into parts (see Fig. 39). Like *Tunnel*, the class *TunnelPart* is derived from *_AbstractTunnel* and inherites all attributes of *_AbstractTunnel*. A *TunnelPart* object should be uniquely related to exactly one tunnel or tunnel part object.

Fig. 39: Example of a tunnel modeled with two tunnel parts (source: Helmut Stracke).

AbstractTunnelType, _AbstractTunnel

```
<xs:complexType name="AbstractTunnelType">
  <xs:complexContent>
    <xs:extension base="core:AbstractSiteType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
```

```
            <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="yearOfConstruction" type="xs:gYear" minOccurs="0"/>
            <xs:element name="yearOfDemolition" type="xs:gYear" minOccurs="0"/>
            <xs:element name="lod1Solid" type="gml:SolidPropertyType" minOccurs="0"/>
            <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod2Solid" type="gml:SolidPropertyType" minOccurs="0"/>
            <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod2MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="outerTunnelInstallation" type="TunnelInstallationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="interiorTunnelInstallation" type="IntTunnelInstallationPropertyType" minOccurs="0"
                        maxOccurs="unbounded"/>
            <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="lod3Solid" type="gml:SolidPropertyType" minOccurs="0"/>
            <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod3MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
            <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod4MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="interiorHollowSpace" type="InteriorHollowSpacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="consistsOfTunnelPart" type="TunnelPartPropertyType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="_GenericApplicationPropertyOfAbstractTunnel" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:element name="_AbstractTunnel" type="AbstractTunnelType" abstract="true" substitutionGroup="core:_Site"/>
    <!-- ========================================================================================== -->
    <xs:element name="_GenericApplicationPropertyOfAbstractTunnel" type="xs:anyType" abstract="true"/>
```

The abstract class *_AbstractTunnel* contains properties for tunnel attributes, purely geometric representations, and geometric/semantic representations of the tunnel or tunnel part in different levels of detail. The attributes describe:

a) The classification of the tunnel or tunnel part (*class*), the different functions (*function*), and the usage (*usage*). The type of these attributes is *gml:CodeType* and the values can be specified in separate code lists.

b) The year of construction (*yearOfConstruction*) and the year of demolition (*yearOfDemolition*) of the tunnel or tunnel part. The *yearOfConstruction* is the year of completion of the tunnel. The *yearOfDemolition* is the year when the demolition of the tunnel was completed. The date (year) refer to real world time (e.g. 2011).

Spanning the different levels of detail, the tunnel model differs in the complexity and granularity of the geometric representation and the thematic structuring of the model into components with a special semantic meaning. This is illustrated in Fig. 40, showing the same tunnel in four different LODs. Some properties of the class *_AbstractTunnel* are also associated with certain LODs.

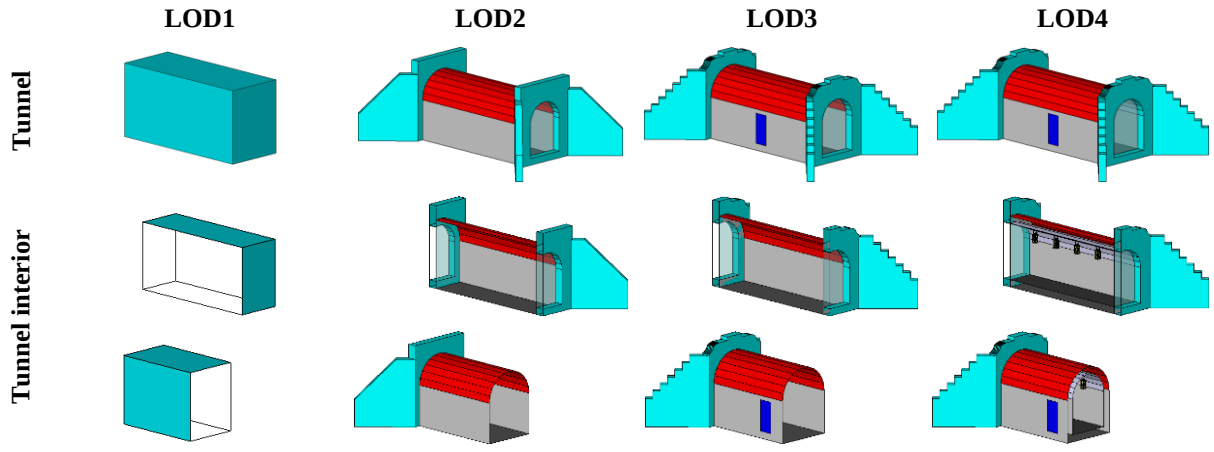

Fig. 40: Tunnel model in LOD1 – LOD4 (source: Karlsruhe Institute of Technology (KIT)).

Tab. 6 shows the correspondence of the different geometric and semantic themes of the tunnel model to LODs. In each LOD, the volume of a tunnel can be expressed by a *gml:Solid* geometry and/or a *gml:MultiSurface*

geometry. The definition of a 3D Terrain Intersection Curve (TIC), used to integrate tunnels from different sources with the Digital Terrain Model, is also possible in all LODs. The TIC can – but does not have to – build closed rings around the tunnel or tunnel parts.

Geometric / semantic theme	**Property type**	**LOD1**	**LOD2**	**LOD3**	**LOD4**
Volume part of the tunnel shell	*gml:SolidType*	•	•	•	•
Surface part of the tunnel shell	*gml:MultiSurfaceType*	•	•	•	•
Terrain intersection curve	*gml:MultiCurveType*	•	•	•	•
Curve part of the tunnel shell	*gml:MultiCurveType*		•	•	•
Tunnel parts	*TunnelPartType*	•	•	•	•
Boundary surfaces (chapter 10.4.3)	*AbstractBoundarySurfaceType*		•	•	•
Outer tunnel installations (chapter 10.4.2)	*TunnelInstallationType*		•	•	•
Openings	*AbstractOpeningType*			•	•
Hollow spaces (chapter 10.4.5)	*HollowSpaceType*				•
Interior tunnel installations	*IntTunnelInstallationType*				•

Tab. 6: Semantic themes of the class *_AbstractTunnel*.

10.4.2 Outer tunnel installations

TunnelInstallationType, TunnelInstallation

A *TunnelInstallation* is an outer component of a tunnel which has not the significance of a *TunnelPart*, but which strongly affects the outer characteristic of the tunnel, for examples stairs. A *TunnelInstallation* optionally has attributes *class*, *function* and *usage*. The attribute *class* - which can only occur once - represents a general classification of the installation. With the attributes *function* and *usage*, nominal and real functions of a tunnel installation can be described. For all three attributes the list of feasible values can specified in a code list. For the geometrical representation of a *TunnelInstallation*, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used. Alternatively, the geometry may be given as *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype tunnel installation is stored only once in a local coordinate system and referenced by other tunnel installation features (see chapter 8.2). The visible surfaces of a tunnel installation can be semantically classified using the concept of boundary surfaces (cf. 10.3.3). A *TunnelInstallation* object should be uniquely related to exactly one tunnel or tunnel part object.

10.4.3 Boundary surfaces

```
<xs:complexType name="AbstractBoundarySurfaceType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="opening" type="OpeningPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfBoundarySurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_BoundarySurface" type="AbstractBoundarySurfaceType" abstract="true" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBoundarySurface" type="xs:anyType" abstract="true"/>
```

_BoundarySurface is the abstract base class for several thematic classes, structuring the exterior shell of a tunnel as well as the visible surface of hollow spaces and both outer and interior tunnel installations. It is a subclass of *_CityObject* and thus inherits all properties like the GML3 standard feature properties (*gml:name* etc.) and the

CityGML specific properties like *ExternalReferences*. From *_BoundarySurface*, the thematic classes *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface*, *ClosureSurface*, *FloorSurface*, *InteriorWallSurface*, and *CeilingSurface* are derived. The thematic classification of tunnel surfaces is illustrated in Fig. 41 for different types of tunnel cross sections and are specified below.

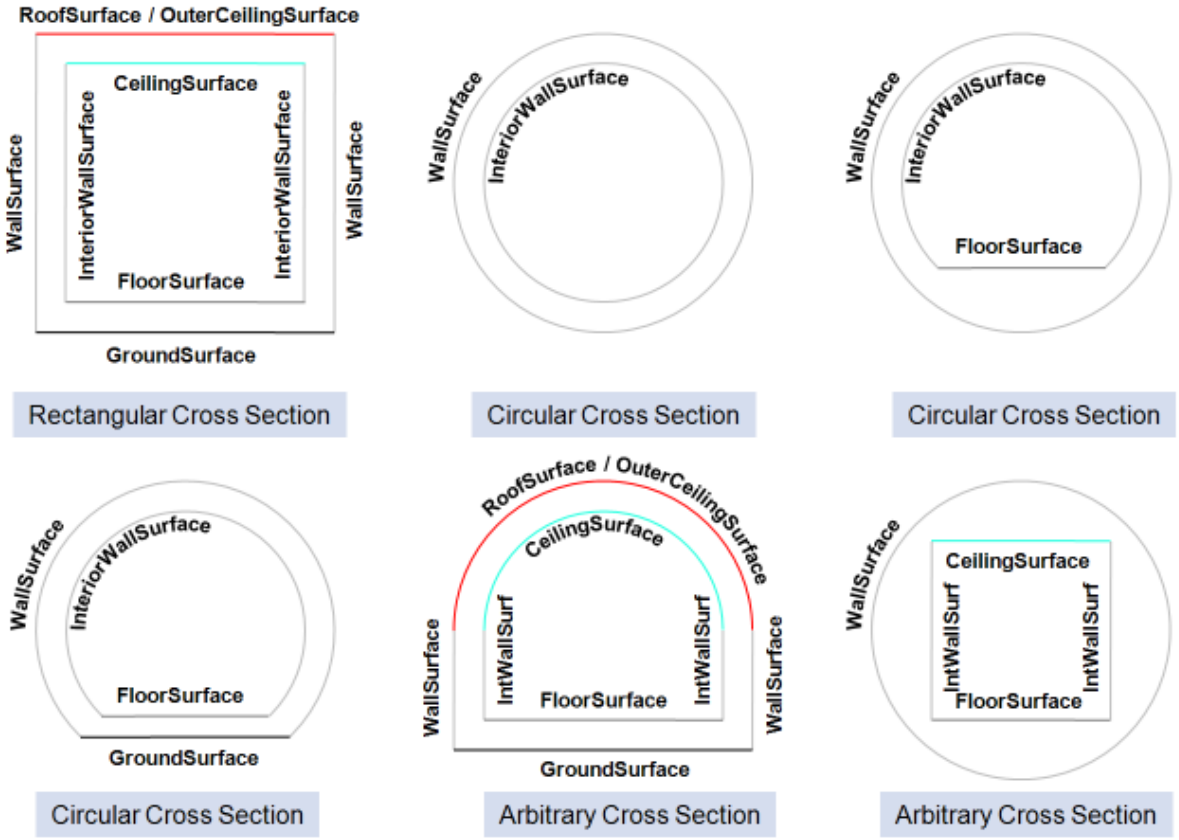

Fig. 41: Examples for the use of boundary surfaces for tunnels with different cross sections. *WallSurface*, *RoofSurface*, *GroundSurface*, *OuterCeilingSurface* and *OuterFloorSurface* are available in LOD2 – 4, whereas *InteriorWallSurface*, *FloorSurface* and *CeilingSurface* may only be used in LOD4 to model the interior boundary surfaces of a hollow space.

For each LOD between 2 and 4, the geometry of a *_BoundarySurface* may be defined by a different *gml:MultiSurface* geometry. Starting from LOD3, a *_BoundarySurface* may contain *_Openings* (cf. chapter 10.4.4) like doors and windows. If the geometric location of openings topologically lies within a surface component (e.g. *gml:Polygon*) of the *gml:MultiSurface* geometry, these openings must be represented as holes within that surface. A hole is represented by an interior ring within the corresponding surface geometry object. According to GML3, the points have to be specified in reverse order (exterior boundaries counter-clockwise and interior boundaries clockwise when looking in opposite direction of the surface's normal vector). If such an opening is sealed by a *Door* or a *Window*, their outer boundary may consist of the same points as the inner ring (denoting the hole) of the surrounding surface. The embrasure surfaces of an opening belong to the relevant adjacent *_BoundarySurface*. If, for example a door seals the opening, the embrasure surface on the one side of the door belongs to the *InteriorWallSurface* and on the other side to the *WallSurface* (cf. right part of Fig. 32 for the same situation in a building model).

GroundSurfaceType, GroundSurface

```
<xs:complexType name="GroundSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
```

```
<xs:element name="GroundSurface" type="GroundSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfGroundSurface" type="xs:anyType" abstract="true"/>
```

The ground plate of a tunnel or tunnel part is modelled by the class *GroundSurface*. Usually a *GroundSurface* is a boundary surface between the tunnel and the surrounding earth (soil, rock etc.) or water.

OuterCeilingSurfaceType, OuterCeilingSurface

```
<xs:complexType name="OuterCeilingSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfOuterCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="OuterCeilingSurface" type="OuterCeilingSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfOuterCeilingSurface" type="xs:anyType" abstract="true"/>
```

A mostly horizontal surface belonging to the outer tunnel shell and with the orientation pointing downwards can be modeled as an *OuterCeilingSurface*. Examples are the visible part of an avalanche protector or the boundary surface between the tunnel and the surrounding earth or water.

WallSurfaceType, WallSurface

```
<xs:complexType name="WallSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWallSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="WallSurface" type="WallSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfWallSurface" type="xs:anyType" abstract="true"/>
```

All parts of the tunnel facade belonging to the outer tunnel shell can be modelled by the class *WallSurface*. Usually a *WallSurface* is a boundary surface between the tunnel and the surrounding earth (soil, rock etc.) or water.

OuterFloorSurfaceType, OuterFloorSurface

```
<xs:complexType name="OuterFloorSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfOuterFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="OuterFloorSurface" type="OuterFloorSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfOuterFloorSurface" type="xs:anyType" abstract="true"/>
```

A mostly horizontal surface belonging to the outer tunnel shell and with the orientation pointing upwards can be modeled as an *OuterFloorSurface*.

RoofSurfaceType, RoofSurface

```
<xs:complexType name="RoofSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfRoofSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="RoofSurface" type="RoofSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfRoofSurface" type="xs:anyType" abstract="true"/>
```

Boundary surfaces belonging to the outer tunnel shell and with the main purpose to protect the tunnel from above are expressed by the class *RoofSurface*. The orientation of these boundaries is mainly pointing upwards.

ClosureSurfaceType, ClosureSurface

```
<xs:complexType name="ClosureSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="ClosureSurface" type="ClosureSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfClosureSurface" type="xs:anyType" abstract="true"/>
```

Openings in tunnels or hollow spaces not filled by a door or a window can be sealed by a virtual surface called *ClosureSurface* (cf. chapter 6.4). For example, the doorways of tunnels can be modelled as *ClosureSurface*.

FloorSurfaceType, FloorSurface

```
<xs:complexType name="FloorSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="FloorSurface" type="FloorSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfFloorSurface" type="xs:anyType" abstract="true"/>
```

The class *FloorSurface* must only be used in the LOD4 interior tunnel model for modelling the floor of hollow spaces.

InteriorWallSurfaceType, InteriorWallSurface

```
<xs:complexType name="InteriorWallSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfInteriorWallSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
<!-- ========================================================================================== -->
<xs:element name="InteriorWallSurface" type="InteriorWallSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfInteriorWallSurface" type="xs:anyType" abstract="true"/>
```

The class *InteriorWallSurface* is only allowed to be used in the LOD4 interior tunnel model for modelling the visible wall surfaces of hollow spaces.

CeilingSurfaceType, CeilingSurface

```
<xs:complexType name="CeilingSurfaceType">
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="CeilingSurface" type="CeilingSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfCeilingSurface" type="xs:anyType" abstract="true"/>
```

The class *CeilingSurface* is only allowed to be used in the LOD4 interior tunnel model for modelling the ceiling of hollow spaces.

10.4.4 Openings

AbstractOpeningType, _Opening

```
<xs:complexType name="AbstractOpeningType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfOpening" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_Opening" type="AbstractOpeningType" abstract="true" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfOpening" type="xs:anyType" abstract="true"/>
```

The class *_Opening* is the abstract base class for semantically describing openings like doors or windows in outer and inner boundary surfaces. Openings only exist in models of LOD3 or LOD4. Each *_Opening* is associated with a *gml:MultiSurface* geometry. Alternatively, the geometry may be given as *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype opening is stored only once in a local coordinate system and referenced by other opening features (see chapter 8.2).

WindowType, Window

```
<xs:complexType name="WindowType">
  <xs:complexContent>
    <xs:extension base="AbstractOpeningType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWindow" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Window" type="WindowType" substitutionGroup="_Opening"/>
```

```
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfWindow" type="xs:anyType" abstract="true"/>
```

The class *Window* is used for modelling windows in the in the exterior shell of a tunnel and in hollow spaces, or hatches between adjacent hollow spaces. The formal difference between the classes *Window* and *Door* is that – in normal cases – Windows are not specifically intended for the transit of people or vehicles.

DoorType, Door

```
<xs:complexType name="DoorType">
  <xs:complexContent>
    <xs:extension base="AbstractOpeningType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfDoor" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Door" type="DoorType" substitutionGroup="_Opening"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfDoor" type="xs:anyType" abstract="true"/>
```

The class *Door* is used for modelling doors in the exterior shell of a tunnel, or between adjacent hollow spaces. Doors can be used by people to enter or leave a tunnel or ahollow space. In contrast to a *ClosureSurface* a door may be closed, blocking the transit of people or vehicles.

10.4.5 Tunnel interior

HollowSpaceType, HollowSpace

```
<xs:complexType name="HollowSpaceType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="interiorFurniture" type="InteriorFurniturePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="hollowSpaceInstallation" type="IntTunnelInstallationPropertyType" minOccurs="0"
                    maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfHollowSpace" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="HollowSpace" type="HollowSpaceType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfHollowSpace" type="xs:anyType" abstract="true"/>
```

A *HollowSpace* is a semantic object for modelling the free space inside a tunnel and should be uniquely related to exactly one tunnel or tunnel part object. It should be closed (if necessary by using *ClosureSurface*) and the geometry normally will be described by a solid (*lod4Solid*). However, if the topological correctness of the boundary cannot be guaranteed, the geometry can alternatively be given as a *MultiSurface* (*lod4MultiSurface*). The surface normals of the outer shell of a GML solid must point outwards. This is important if appearences should be assigned to *HollowSpace* surfaces. In this case, textures and colors must be placed on the backside of the corresponding surfaces in order to be visible from the inside of the hollow space.

In addition to the geometrical representation, different parts of the visible surface of a hollow space can be modelled by specialised boundary surfaces (*FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface*, cf. chapter 10.4.3).

TunnelFurnitureType, TunnelFurniture

```
<xs:complexType name="TunnelFurnitureType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfTunnelFurniture" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="TunnelFurniture" type="TunnelFurnitureType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTunnelFurniture" type="xs:anyType" abstract="true"/>
```

Hollow spaces may have *TunnelFurniture*. A *TunnelFurniture* is a movable part of a hollow space. A *TunnelFurniture* object should be uniquely related to exactly one hollow space. Its geometry may be represented by an explicit geometry or an *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype tunnel furniture is stored only once in a local coordinate system and referenced by other tunnel furniture features (see chapter 8.2).

IntTunnelInstallationType, IntTunnelInstallation

```
<xs:complexType name="IntTunnelInstallationType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfIntTunnelInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="IntTunnelInstallation" type="IntTunnelInstallationType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfIntTunnelInstallation" type="xs:anyType" abstract="true"/>
```

An *IntTunnelInstallation* is an object inside a tunnel with a specialized function or semantic meaning. In contrast to *TunnelFurniture*, objects of the class *IntTunnelInstallation* are permanently attached to the tunnel structure and cannot be moved. Typical examples are interior stairs, railings, radiators or pipes. Objects of the class *IntTunnelInstallation* can either be associated with a hollow space (class *HollowSpace*), or with the complete tunnel or tunnel part (class *_AbstractTunnel*, see chapter 10.4.1). However, they should be uniquely related to exactly one hollow space or one tunnel / tunnel part object. An *IntTunnelInstallation* optionally has the attributes *class*, *function* and *usage*. The attribute *class*, which can only occur once, represents a general classification of the internal tunnel component. With the attributes *function* and *usage*, nominal and real functions of a tunnel installation can be described. For all three attributes the list of feasible values can be specified in a code list. For the geometrical representation of an *IntTunnelInstallation*, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used. Alternatively, the geometry may be given as *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype interior tunnel installation is stored only once in a local coordinate system and referenced by other interior tunnel installation features (see chapter 8.2). The visible surfaces of an interior tunnel installation can be semantically classified using the concept of boundary surfaces (cf. 10.4.3).

10.4.6 Examples

The example in Fig. 42 shows a pedestrian underpass in the city centre of Karlsruhe, Germany. On the left side of Fig. 42, a photo illustrates the real world situation. Both entrances of the underpass are marked in the photo by dashed rectangles. On the right side of the figure, the CityGML tunnel model is shown. The terrain surrounding the tunnel has been virtually cut out of model in order to visualize the entire tunnel with its subsurface body. The same underpass is illustrated in Fig. 43 from a different perspective. The camera is positioned in front of the left entrance (black dashed rectangle in Fig. 42) and pointing in the direction of the right entrance (white dashed rectangle in Fig. 42). On the right side of Fig. 43, the tunnel model is shown from the same perspective. Again holes are cut in the terrain surface in order to make the subsurface part of the tunnel visible. An LOD1 representation of the nearby buildings is shown in the background of the model.

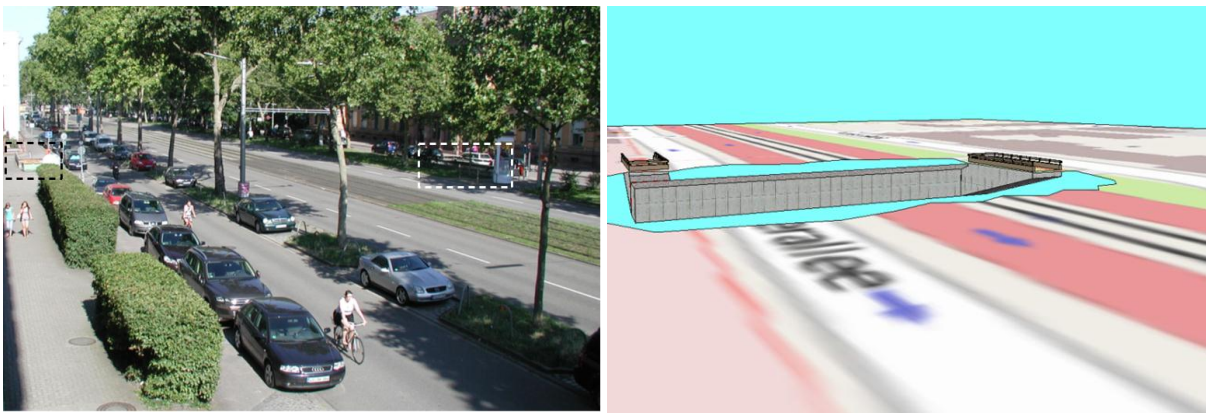

Fig. 42: Example of a tunnel modeled in LOD3 (real situation on the left side; CityGML model on the right side) (source: Karlsruhe Institute of Technology (KIT), courtesy of City of Karlsruhe).

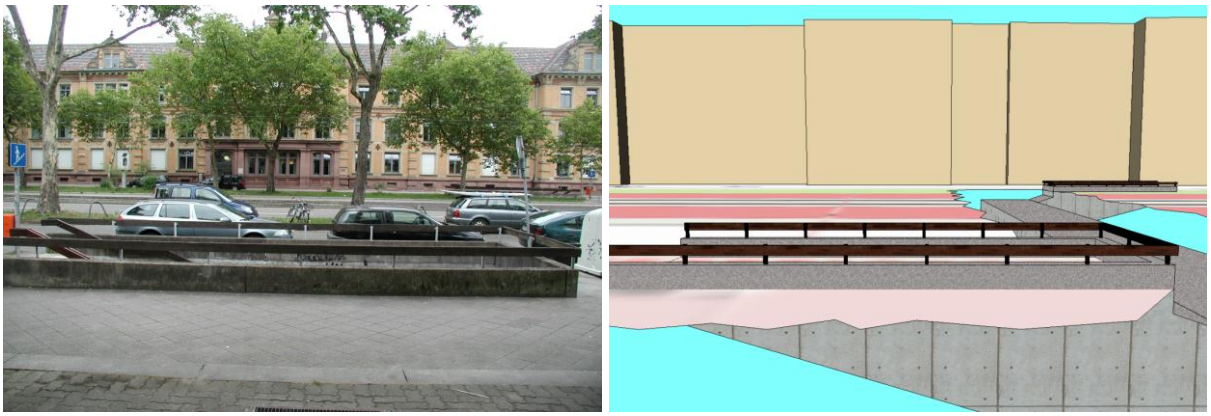

Fig. 43: The same LOD3 tunnel shown from a different perspective. The camera is positioned in front of the left entrance and pointing in the direction of the right entrance. (real situation on the left side; CityGML model on the right side). The model on the right also includes an LOD1 representation of the nearby buildings in the background (painted in light brown) (source: Karlsruhe Institute of Technology (KIT), courtesy of City of Karlsruhe).

The model is subdivided into one *Tunnel* (the actual underpass) and two *TunnelParts* (both entrances). The tunnel and tunnel parts are bounded by *GroundSurface*, *WallSurface*, *RoofSurface*. *ClosureSurface* objects are used to virtually seal the tunnel entrances. For safety reasons each of the two entrances has railings which are modeled as *TunnelInstallation*. Due to the high geometrical accuracy and the semantic richness, the model is classified as LOD3.

10.4.7 Code lists

The attributes *class*, *function*, and *usage* of the features *_AbstractTunnel*, *TunnelInstallation*, *HollowSpace*, *TunnelFurniture* and *IntTunnelInstallation* are specified as *gml:CodeType*. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.2.

10.4.8 Conformance requirements

Base requirements

1. If a tunnel only consists of one (homogeneous) part, it shall be represented by the element *Tunnel*. However, if a tunnel is composed of individual structural segments, it shall be modelled as a *Tunnel* element having one or more additional *TunnelPart* elements. Only the geometry and non-spatial properties of the main part of the tunnel should be represented within the aggregating *Tunnel* element.

Usage restriction of tunnel model components according to different LODs

2. The *lodXSolid* and *lodXMultiSurface*, X ∈ [1..4], properties (*gml:SolidPropertyType* resp. *gml:MultiSurfacePropertyType*) of *_AbstractTunnel* may be used to geometrically represent the exterior shell of a tunnel (as volume or surface model) within each LOD. For LOD1, either *lod1Solid* or *lod1MultiSurface* must be used, but not both. Starting from LOD2, both properties may be modelled individually and complementary.
3. Starting from LOD2, the exterior shell of an *_AbstractTunnel* may be semantically decomposed into *_BoundarySurface* elements using the *boundedBy* property (type: *BoundarySurfacePropertyType*) of *_AbstractTunnel*. Only *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface* and *ClosureSurface* as subclasses of *_BoundarySurface* are allowed. The *boundedBy* property (not to be confused with the *gml:boundedBy* property) shall not be used if the tunnel is only represented in LOD1.

   If the exterior shell is represented by *_BoundarySurface* elements, an additional geometric representation as volume or surface model using the *lodXSolid* and *lodXMultiSurface*, X ∈ [2..4], properties shall not explicitly define the geometry, but has to reference the according components of the *gml:MultiSurface* element of *_BoundarySurface* within each LOD using the XLink concept of GML 3.1.1.
4. Starting from LOD2, curve parts of the tunnel shell may be represented using the *lodXMultiCurve*, X ∈ [2..4], property of *_AbstractTunnel*. This property shall not be used if the tunnel is only represented in LOD1.
5. Starting from LOD2, the *outerTunnelInstallation* property (type: *TunnelInstallationPropertyType*) of *_AbstractTunnel* may be used to model *TunnelInstallation* elements. *TunnelInstallation* elements shall only be used to represent outer characteristics of a tunnel which do not have the significance of tunnel parts. The *outerTunnelInstallation* property shall not be used if the tunnel is only represented in LOD1.
6. Starting from LOD2, the geometry of *TunnelInstallation* elements may be semantically classified by *_BoundarySurface* elements using the *boundedBy* property (type: *BoundarySurfacePropertyType*) of *TunnelInstallation*. Only *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface* and *ClosureSurface* as subclasses of *_BoundarySurface* are allowed.
7. Starting from LOD3, openings of *_BoundarySurface* elements may be modelled using the opening property (type: *OpeningPropertyType*) of *_BoundarySurface*. This property shall not be used for *_BoundarySurface* elements only represented in LOD2. Accordingly, the surface geometry representing a *_BoundarySurface* in LOD2 must be simply connected.

   The opening property of *_BoundarySurface* may contain or reference *_Opening* elements. If the geometric location of an *_Opening* element topologically lies within a surface component of the *_BoundarySurface*, the opening must also be represented as inner hole of that surface. The embrasure surface of an *_Opening* element shall belong to the relevant adjacent *_BoundarySurface*.
8. Starting from LOD4, the *interiorHollowSpace* property (type: *InteriorHollowSpacePropertyType*) of *_AbstractTunnel* may be used to semantically model the free space inside the tunnel by *HollowSpace* elements. This property shall not be used if the tunnel is only represented in LOD 1 – 3. The *HollowSpace* element may be geometrically represented as a surface or volume model, using its *lod4Solid* or *lod4MultiSurface* property (*gml:SolidPropertyType* resp. *gml:MultiSurfacePropertyType*).

   In addition, different parts of the visible surface of a hollow space may be modelled by thematic *_BoundarySurface* elements. Only *FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface* as subclasses of *_BoundarySurface* are allowed. If the visible surface of a hollow space is repre-

sented by *_BoundarySurface* elements, an additional geometric representation as volume or surface model using the *lod4Solid* and *lod4MultiSurface* property shall not explicitly define the geometry, but has to reference the according components of the *gml:MultiSurface* element of *_BoundarySurface* using the XLink concept of GML 3.1.1.

9. Starting from LOD4, the *interiorTunnelInstallation* property (type: *IntTunnelInstallationPropertyType*) of *_AbstractTunnel* may be used to represent immovable objects inside the tunnel that are permanently attached to the tunnel structure. The *interiorTunnelInstallation* property shall not be used if the tunnel is only represented in LOD 1 – 3. Furthermore, the *interiorTunnelInstallation* property shall only be used if the object cannot be associated with a *HollowSpace* element. In the latter case, the *hollowSpaceInstallation* property (type: *IntTunnelInstallationPropertyType*) of the corresponding *HollowSpace* element shall be used to represent the object.
10. Starting from LOD4, the geometry of *IntTunnelInstallation* elements may be semantically classified by *_BoundarySurface* elements using the *boundedBy* property (type: *BoundarySurfacePropertyType*) of *IntTunnelInstallation*. Only *FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface* as subclasses of *_BoundarySurface* are allowed.

Referential integrity

11. The *boundedBy* property (type: *BoundarySurfacePropertyType*) of the element *_AbstractTunnel* may contain a *_BoundarySurface* element inline or an XLink reference to a remote *_BoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_BoundarySurface* element (where remote *_BoundarySurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

    Only *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface* and *ClosureSurface* elements are allowed to be encapsulated or referenced by the boundedBy property of _AbstractTunnel.
12. The *outerTunnelInstallation* property (type: *TunnelInstallationPropertyType*) of the element *_AbstractTunnel* may contain a *TunnelInstallation* element inline or an XLink reference to a remote *TunnelInstallation* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *outerTunnelInstallation* property may only point to a remote *TunnelInstallation* element (where remote *TunnelInstallation* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
13. The *interiorTunnelInstallation* property (type: *IntTunnelInstallationPropertyType*) of the element *_AbstractTunnel* may contain an *IntTunnelInstallation* element inline or an XLink reference to a remote *IntTunnelInstallation* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *interiorTunnelInstallation* property may only point to a remote *IntTunnelInstallation* element (where remote *IntTunnelInstallation* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
14. The *interiorHollowSpace* property (type: *InteriorHollowSpacePropertyType*) of the element *_AbstractTunnel* may contain a *HollowSpace* element inline or an XLink reference to a remote *HollowSpace* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *interiorHollowSpace* property may only point to a remote *HollowSpace* element (where remote *HollowSpace* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
15. The *consistsOfTunnelPart* property (type: *TunnelPartPropertyType*) of the element *_AbstractTunnel* may contain a *TunnelPart* element inline or an XLink reference to a remote *TunnelPart* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *consistsOfTunnelPart* property may only point to a remote *TunnelPart* element (where remote *TunnelPart* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
16. The *opening* property (type: *OpeningPropertyType*) of the element *_BoundarySurface* may contain an *_Opening* element inline or an XLink reference to a remote *_Opening* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the opening property may only point to a re-

mote *_Opening* element (where remote *_Opening* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

17. The *boundedBy* property (type: *BoundarySurfacePropertyType*) of the element *TunnelInstallation* may contain a *_BoundarySurface* element inline or an XLink reference to a remote *_BoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_BoundarySurface* element (where remote *_BoundarySurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

    Only *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface* and *ClosureSurface* elements are allowed to be encapsulated or referenced by the *boundedBy* property of *TunnelInstallation*.

18. The *boundedBy* property (type: *BoundarySurfacePropertyType*) of the element *IntTunnelInstallation* may contain a *_BoundarySurface* element inline or an XLink reference to a remote *_BoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_BoundarySurface* element (where remote *_BoundarySurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

    Only *FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface* elements are allowed to be encapsulated or referenced by the *boundedBy* property of *IntTunnelInstallation*.

19. The *boundedBy* property (type: *BoundarySurfacePropertyType*) of the element *HollowSpace* may contain a *_BoundarySurface* element inline or an XLink reference to a remote *_BoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_BoundarySurface* element (where remote *_BoundarySurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

    Only *FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface* elements are allowed to be encapsulated or referenced by the *boundedBy* property of *HollowSpace*.

20. The *interiorFurniture* property (type: *InteriorFurniturePropertyType*) of the element *HollowSpace* may contain an *TunnelFurniture* element inline or an XLink reference to a remote *TunnelFurniture* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *interiorFurniture* property may only point to a remote *TunnelFurniture* element (where remote *TunnelFurniture* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

21. The *hollowSpaceInstallation* property (type: *IntTunnelInstallationPropertyType*) of the element *HollowSpace* may contain an *IntTunnelInstallation* element inline or an XLink reference to a remote *IntTunnelInstallation* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *hollowSpaceInstallation* property may only point to a remote *IntTunnelInstallation* element (where remote *IntTunnelInstallation* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

22. The *lodXImplicitRepresentation*, $X \in [2..4]$, property (type: *core:ImplicitRepresentationPropertyType*) of the element *TunnelInstallation* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lodXImplicitRepresentation*, $X \in [2..4]$, property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

23. The *lod4ImplicitRepresentation* property (type: *core:ImplicitRepresentationPropertyType*) of the element *IntTunnelInstallation* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lod4ImplicitRepresentation* property may only point to a remote

core:ImplicitGeometry element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

24. The *lodXImplicitRepresentation*, X ∈ [3..4], property (type: *core:ImplicitRepresentationPropertyType*) of the element *_Opening* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lodXImplicitRepresentation*, X ∈ [3..4], property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
25. The *lod4ImplicitRepresentation* property (type: *core:ImplicitRepresentationPropertyType*) of the element *TunnelFurniture* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lod4ImplicitRepresentation* property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

10.5 Bridge model

The bridge model allows for the representation of the thematic, spatial and visual aspects of bridges and bridge parts in four levels of detail, LOD 1 – 4. The bridge model of CityGML is defined by the thematic extension module *Bridge* (cf. chapter 7). Fig. 44 illustrates examples of bridge models in all LODs.

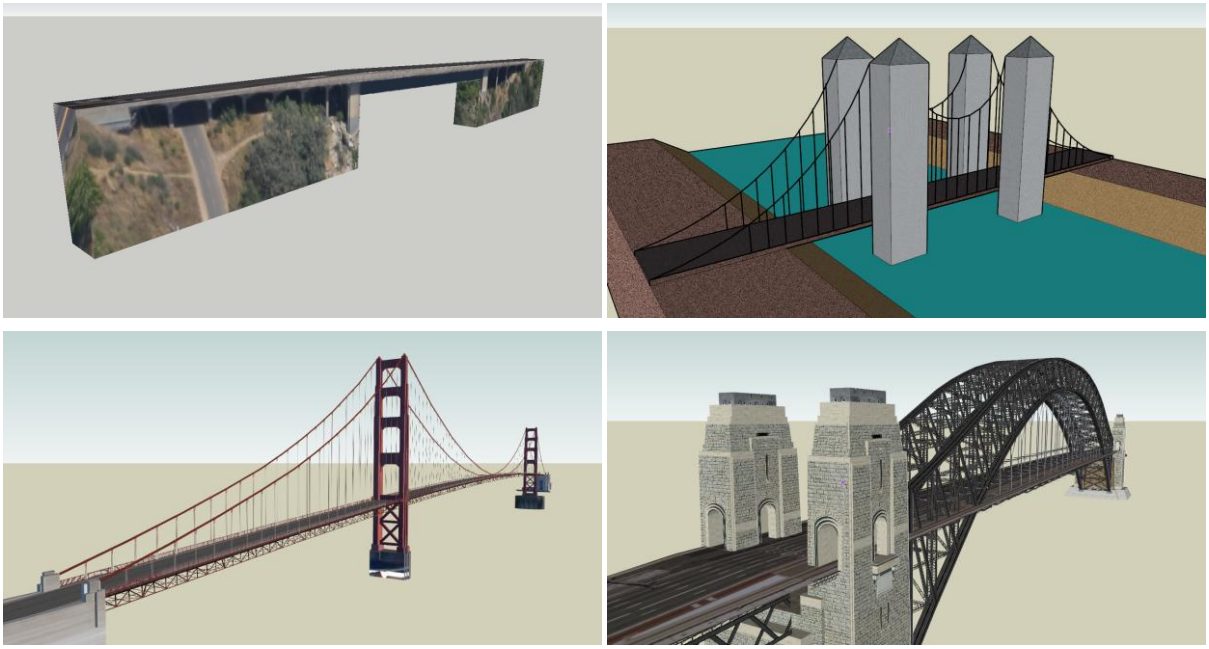

Fig. 44: Examples for bridge models in LOD1 (upper left), LOD2 (upper right), LOD3 (lower left) and LOD4 (lower right) (source: Google 3D warehouse)

The bridge model was developed in analogy to the building model (cf. chapter 10.3) with regard to structure and attributes. The UML diagram of the bridge model is depicted in Fig. 45, and the XML schema definition is presented in annex A.3.

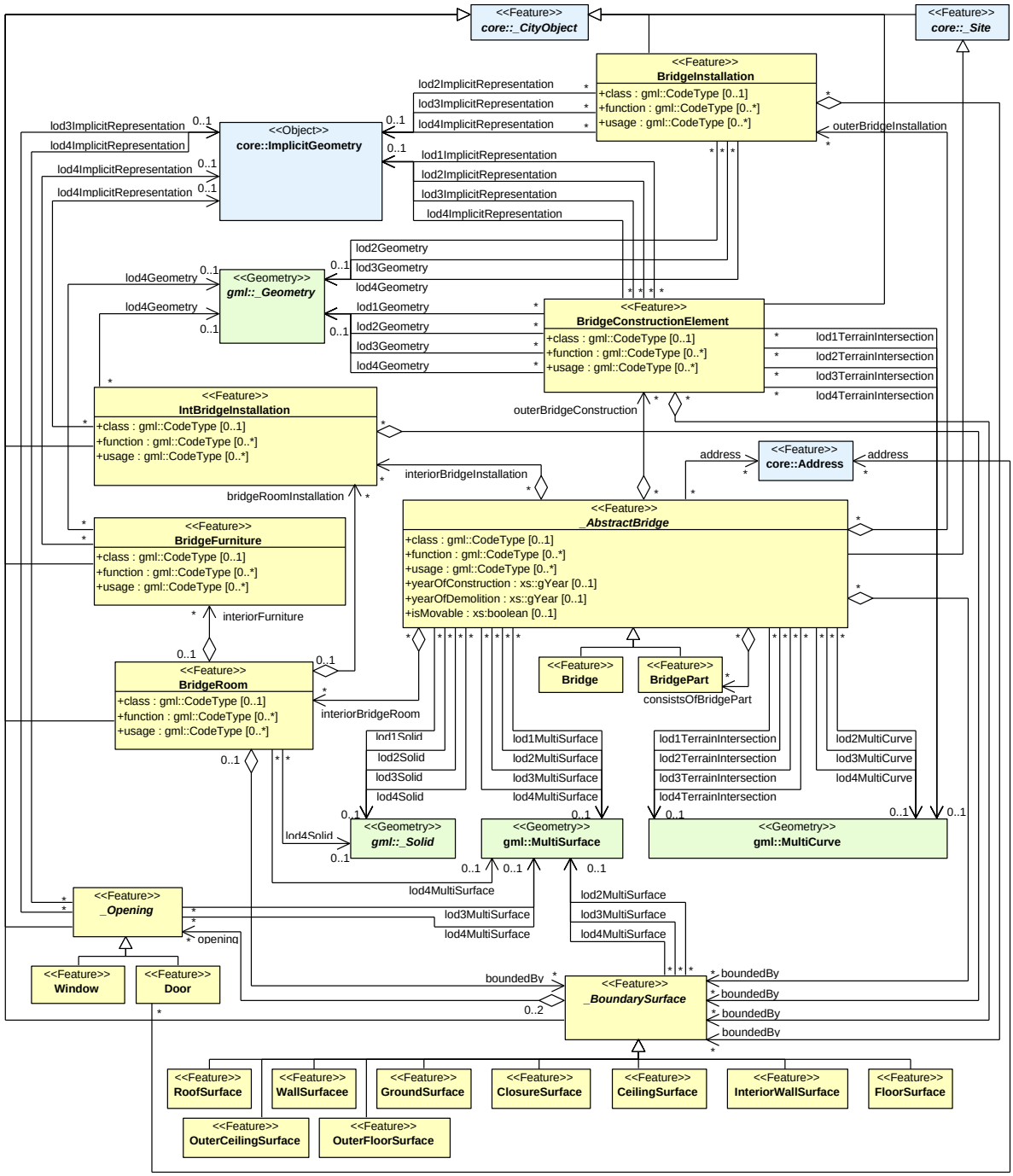

Fig. 45: UML diagram of the bridge model, part one.

A (movable or unmovable) bridge is represented by an object of the class *Bridge*. This class inherits its attributes and relations from the abstract base class *_AbstractBridge*. The spatial properties are defined by a solid for each of the four LODs (relations *lod1Solid* to *lod4Solid*). In analogy to the building model, the semantical as well as the geometrical richness increases from LOD1 (blocks model) to LOD3 (architectural model). Simple examples of bridges in each of those LODs are depicted in Fig. 46. Interior structures like rooms are dedicated to LOD4. To cover the case of bridge models where the topology does not satisfy the properties of a solid (essentially water tightness), a multi surface representation is allowed (*lod1MultiSurface* to *lod4MultiSurface*). The line where the bridge touches the terrain surface is represented by a terrain intersection curve, which is provided for each LOD (relations *lod1TerrainIntersection* to *lod4TerrainIntersection*). In addition to the solid representation of a bridge, linear characteristics like ropes or antennas can be specified geometrically by the *lod1MultiCurve* to *lod4MultiCurve* relations. If those characteristics shall be represented semantically, the features *BridgeInstallation* or *BridgeConstructionElement* can be used (see section 10.5.2). All relations to semantic objects and geometric properties are listed in Tab. 7.

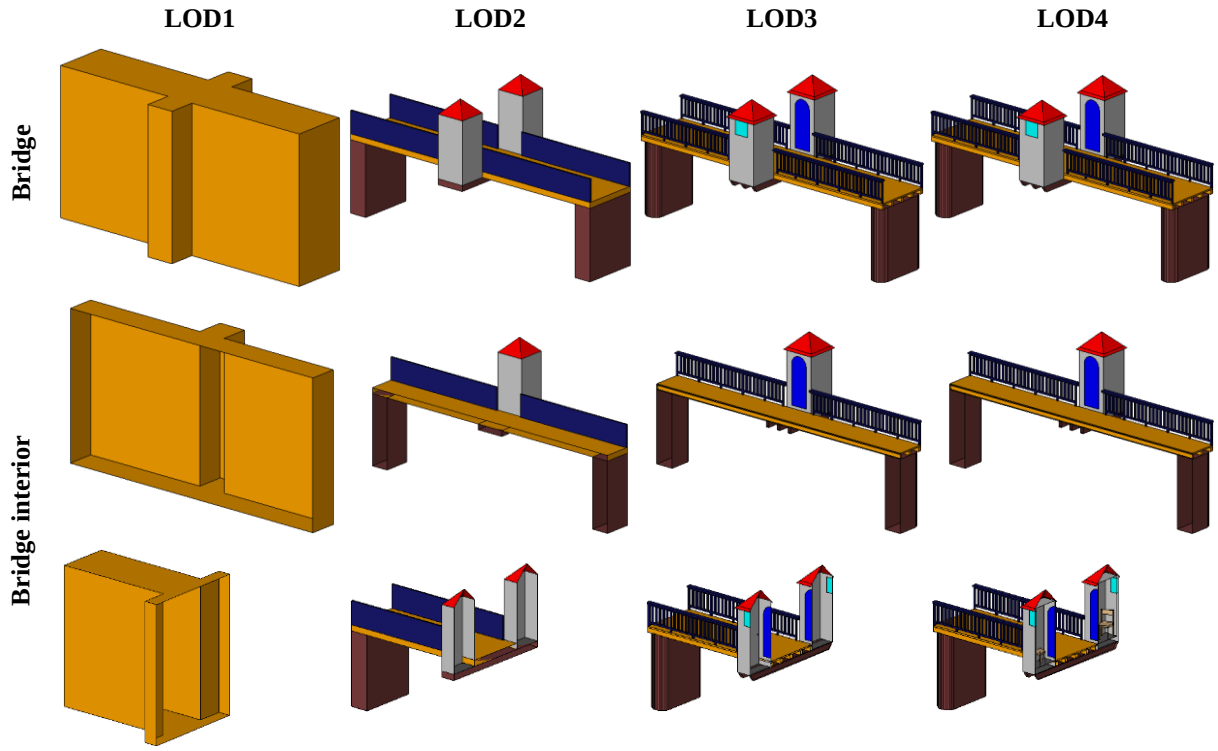

Fig. 46: Bridge model in LOD1 – LOD4. (source: Karlsruhe Institute of Technology (KIT))

The semantic attributes of an *_AbstractBridge* are *class*, *function*, *usage* and *is_movable*. The attribute *class* is used to classify bridges, e.g. to distinguish different construction types (cf. Fig. 48). The attribute *function* allows representing the utilization of the bridge independently of the construction. Possible values may be railway bridge, roadway bridge, pedestrian bridge, aqueduct, etc. The option to denote a usage which is divergent to one of the primary functions of the bridge (*function*) is given by the attribute *usage*. The type of these attributes is *gml:CodeType*, the values of which can be defined in code lists. The name of the bridge can be represented by the *gml:name* attribute, which is inherited from the base class *gml:_GML* via the classes *gml:_Feature*, *_CityObject*, and *_Site*. Each *Bridge* or *BridgePart* feature may be assigned zero or more addresses using the *address* property. The corresponding *AddressPropertyType* is defined within the CityGML core module (cf. chapter 10.1.4).

Geometric / semantic theme	**Property type**	**LOD1**	**LOD2**	**LOD3**	**LOD4**
Volume part of the bridge shell	*gml:SolidType*	•	•	•	•
Surface part of the bridge shell	*gml:MultiSurfaceType*	•	•	•	•
Terrain intersection curve	*gml:MultiCurveType*	•	•	•	•
Curve part of the bridge shell	*gml:MultiCurveType*		•	•	•
Bridge parts (chapter 10.5.1)	*BridgePartType*	•	•	•	•
Boundary surfaces (chapter 10.5.3)	*AbstractBoundarySurfaceType*		•	•	•
Outer bridge installations (chapter 10.5.2)	*BridgeInstallationType*		•	•	•
Bridge construction elements (chapter 10.5.2)	*BridgeConstruction-ElementType*	•	•	•	•
Openings (chapter 10.5.4)	*AbstractOpeningType*			•	•
Bridge rooms (chapter 10.5.5)	*BridgeRoomType*				•
Interior bridge installations	*IntBridgeInstallationType*				•

Tab. 7: Semantic themes of the class *_AbstractBridge*.

The boolean attribute *is_movable* is defined to specify whether a bridge is movable or not. The modeling of the geometric aspects of the movement is delayed to later versions of this standard. Some types of movable bridges are depicted in Fig. 47.

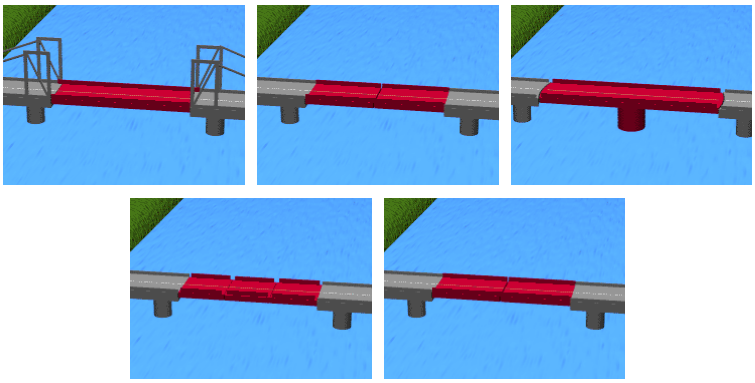

Fig. 47: Examples for movable bridges (source: ISO 6707).

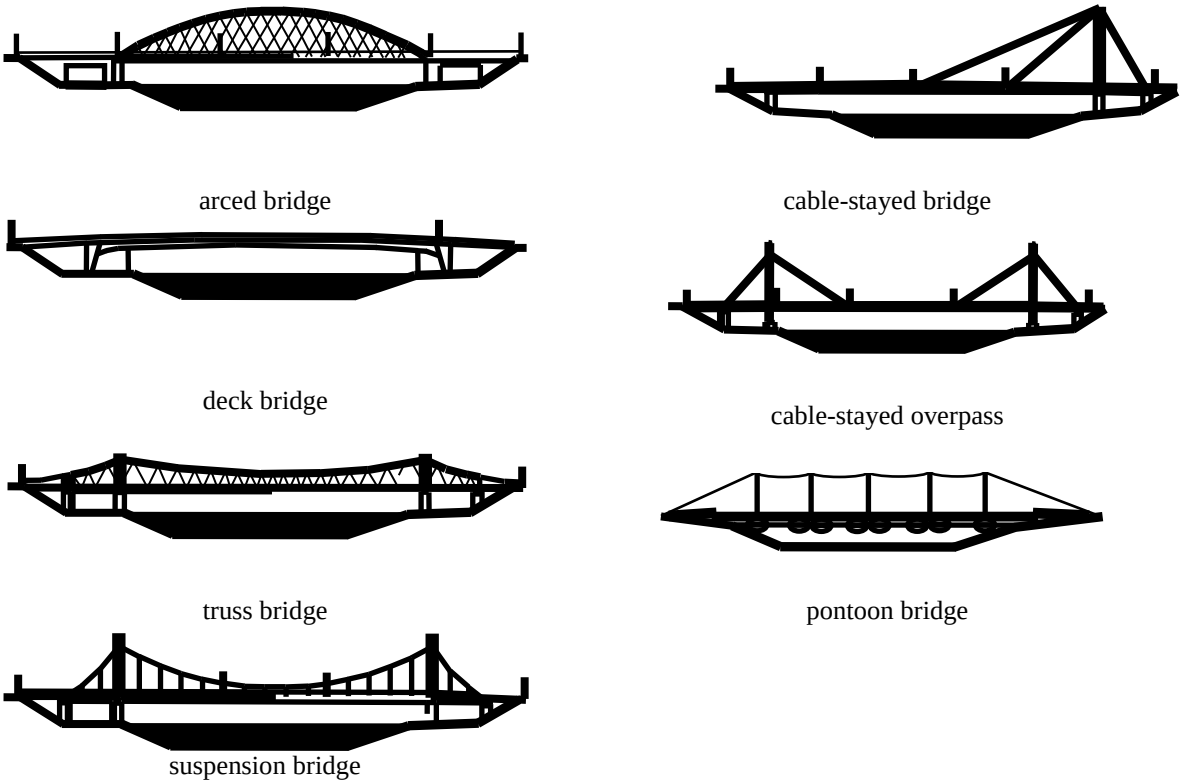

Fig. 48: Examples for different types of bridges.

XML namespace

The XML namespace of the CityGML Bridge module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/bridge/2.0. Within the XML Schema definition of the Bridge module, this URI is also used to identify the default namespace.

10.5.1 Bridge and bridge part

BridgeType, Bridge

```
<xs:complexType name="BridgeType">
  <xs:complexContent>
    <xs:extension base="AbstractBridgeType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfBridge" minOccurs="0" maxOccurs="unbounded"/>
```

```
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Bridge" type="BridgeType" substitutionGroup="_AbstractBridge"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBridge" type="xs:anyType" abstract="true"/>
```

BridgePartType, BridgePart

```
<xs:complexType name="BridgePartType">
  <xs:complexContent>
    <xs:extension base="AbstractBridgeType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfBridgePart" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="BridgePart" type="BridgePartType" substitutionGroup="_AbstractBridge"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBridgePart" type="xs:anyType" abstract="true"/>
```

If some parts of a bridge differ from the remaining bridge with regard to attribute values or if parts like ramps can be identified as objects of their own, those parts can be represented as *BridgePart*. A bridge can consist of multiple *BridgeParts*. Like *Bridge*, *BridgePart* is a subclass of *_AbstractBridge* and hence, has the same attributes and relations. The relation *consistOfBridgePart* represents the aggregation hierarchy between a *Bridge* (or a *BridgePart*) and it's *BridgeParts*. By this means, an aggregation hierarchy of arbitrary depth can be modeled. Each *BridgePart* belongs to exactly one *Bridge* (or *BridgePart*). Similar to the building model, the aggregation structure of a bridge forms a tree. A simple example for a bridge with parts is a twin bridge. Another example is presented in chapter 10.5.6.

AbstractBridgeType, _AbstractBridge

```
<xs:complexType name="AbstractBridgeType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractSiteType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="yearOfConstruction" type="xs:gYear" minOccurs="0"/>
        <xs:element name="yearOfDemolition" type="xs:gYear" minOccurs="0"/>
        <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
        <xs:element name="lod1Solid" type="gml:SolidPropertyType" minOccurs="0"/>
        <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod2Solid" type="gml:SolidPropertyType" minOccurs="0"/>
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod2MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="outerBridgeConstruction" type="BridgeConstructionElementPropertyType" minOccurs="0"
                    maxOccurs="unbounded"/>
        <xs:element name="outerBridgeInstallation" type="BridgeInstallationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="interiorBridgeInstallation" type="IntBridgeInstallationPropertyType" minOccurs="0"
                    maxOccurs="unbounded"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod3Solid" type="gml:SolidPropertyType" minOccurs="0"/>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="interiorBridgeRoom" type="InteriorBridgeRoomPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="consistsOfBridgePart" type="BridgePartPropertyType" minOccurs="0" maxOccurs="unbounded"/>
```



```
        <xs:element name="address" type="core:AddressPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfAbstractBridge" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_AbstractBridge" type="AbstractBridgeType" abstract="true" substitutionGroup="core:_Site"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfAbstractBridge" type="xs:anyType" abstract="true"/>
```

The abstract class *_AbstractBridge* is the base class of *Bridges* and *BridgeParts*. It contains properties for bridge attributes, purely geometric representations, and geometric/semantic representations of the bridge or bridge part in different levels of detail. The attributes describe:

a) The classification of the bridge or bridge part (*class*), the different intended usages (*function*), and the different actual usages (*usage*). The permitted values for these property types can be specified in code lists.

b) The year of construction (*yearOfConstruction*) and the year of demolition (*yearOfDemolition*) of the bridge or bridge part. These attributes can be used to describe the chronology of the bridge development within a city model. The points of time refer to real world time.

c) Wether the bridge is movable is specified by the Boolean attribute *isMovable*.

10.5.2 Bridge construction elements and bridge installations

BridgeConstructionElementType, BridgeConstructionElement

```
<xs:complexType name="BridgeConstructionElementType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfBridgeConstructionElement" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="BridgeConstructionElement" type="BridgeConstructionElementType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBridgeConstructionElement" type="xs:anyType"/>
```

BridgeInstallationType, BridgeInstallation

```
<xs:complexType name="BridgeInstallationType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
```

```
        <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfBridgeInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="BridgeInstallation" type="BridgeInstallationType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfBridgeInstallation" type="xs:anyType" abstract="true"/>
```

Bridge elements which do not have the size, significance or meaning of a *BridgePart* can be modelled either as *BridgeConstructionElement* or as *BridgeInstallation*. Elements which are essential from a structural point of view are modelled as *BridgeConstructionElement*, for example structural elements like pylons, anchorages etc. (cf. Fig. 49). A general classification as well as the intended and actual function of the construction element are represented by the attributes *class*, *function*, and *usage*. The geometry of a *BridgeConstructionElement*, which may be present in LOD1 to LOD4, is *gml:_Geometry*. Alternatively, the geometry may be given as *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype bridge construction element is stored only once in a local coordinate system and referenced by other bridge construction element features (cf. chapter 8.2). The visible surfaces of a bridge construction element can be semantically classified using the concept of boundary surfaces (cf. chapter 10.5.3).

Whereas a *BridgeConstructionElement* has structural relevance, a *BridgeInstallation* represents an element of the bridge which can be eliminated without collapsing of the bridge (e.g. stairway, antenna, railing). *BridgeInstallations* occur in LOD 2 to 4 only and are geometrically representated as *gml:_Geometry*. Again, the concept of *ImplicitGeometry* can be applied to *BridgeInstallations* alternatively, and their visible surfaces can be semantically classified using the concept of boundary surfaces (cf. chapter 10.5.3). The class *BridgeInstallation* contains the semantic attributes *class*, *function* and *usage*. The attribute *class* gives a classification of installations of a bridge. With the attributes *function* and *usage*, nominal and real functions of the bridge installation can be described. The type of all attributes is *gml:CodeType* and their values can be defined in code lists.

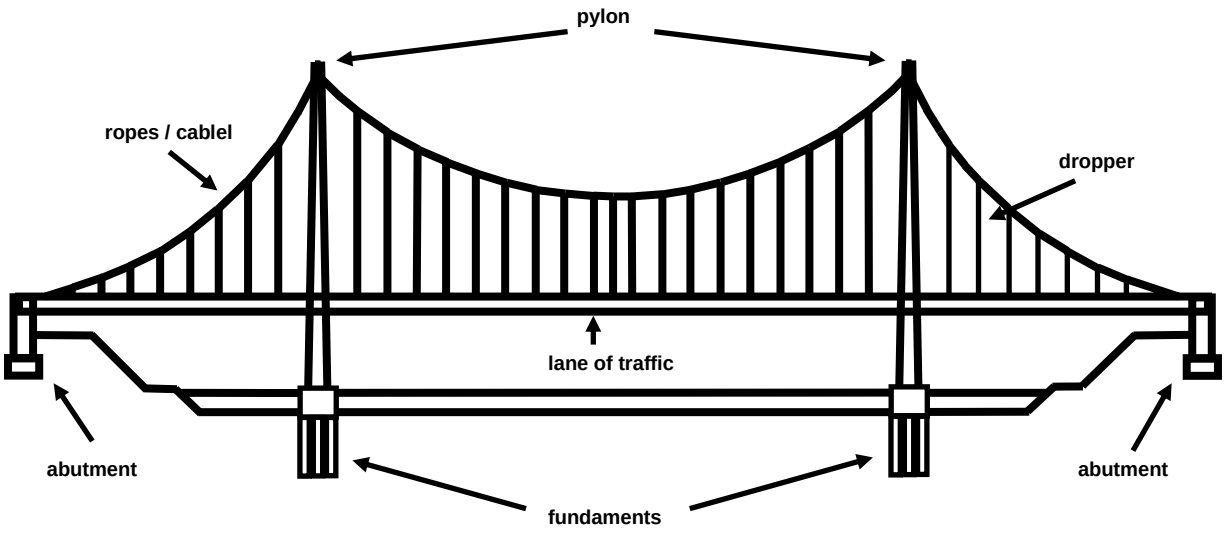

Fig. 49: BridgeConstructionElements of a suspension bridge.

10.5.3 Boundary surfaces

AbstractBoundarySurfaceType, _BoundarySurface

```
<xs:complexType name="AbstractBoundarySurfaceType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="opening" type="OpeningPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfBoundarySurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_BoundarySurface" type="AbstractBoundarySurfaceType" abstract="true" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBoundarySurface" type="xs:anyType" abstract="true"/>
```

The thematic boundary surfaces of a bridge are defined in analogy to the building module. *_BoundarySurface* is the abstract base class for several thematic classes, structuring the exterior shell of a bridge as well as the visible surfaces of rooms, bridge construction elements and both outer and interior bridge installations. It is a subclass of *_CityObject* and thus inherits all properties like the GML3 standard feature properties (*gml:name* etc.) and the CityGML specific properties like *ExternalReferences*. From *_BoundarySurface*, the thematic classes *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface*, *ClosureSurface*, *FloorSurface*, *InteriorWallSurface*, and *CeilingSurface* are derived.

For each LOD between 2 and 4, the geometry of a *_BoundarySurface* may be defined by a different *gml:MultiSurface* geometry.

In LOD3 and LOD4, a *_BoundarySurface* may contain *_Openings* (cf. chapter 10.5.4) like doors and windows. If the geometric location of *_Openings* topologically lies within a surface component (e.g. *gml:Polygon*) of the *gml:MultiSurface* geometry, these *_Openings* must be represented as holes within that surface. A hole is represented by an interior ring within the corresponding surface geometry object. According to GML3, the points have to be specified in reverse order (exterior boundaries counter-clockwise and interior boundaries clockwise when looking in opposite direction of the surface's normal vector). If such an opening is sealed by a *Door*, a *Window*, or a *ClosureSurface*, their outer boundary may consist of the same points as the inner ring (denoting the hole) of the surrounding surface. The embrasure surfaces of an *Opening* belong to the relevant adjacent *_BoundarySurface*. If, for example a door seals the *Opening*, the embrasure surface on the one side of the door belongs to the *InteriorWallSurface* and on the other side to the *WallSurface.*

Fig. 50 depicts a bridge with *RoofSurfaces*, *WallSurfaces*, *OuterFloorSurfaces* and *OuterCeilingSurfaces*. Besides *Bridges* and *BridgeParts*, *BridgeConstructionElements*, *BridgeInstallations* as well as *IntBridgeInstallations* can be related to *_BoundarySurface*. *_BoundarySurfaces* occur in LOD2 to LOD4. In LOD3 and LOD4, such a surface may contain *_Openings* (see chapter 10.3.4) like doors and windows.

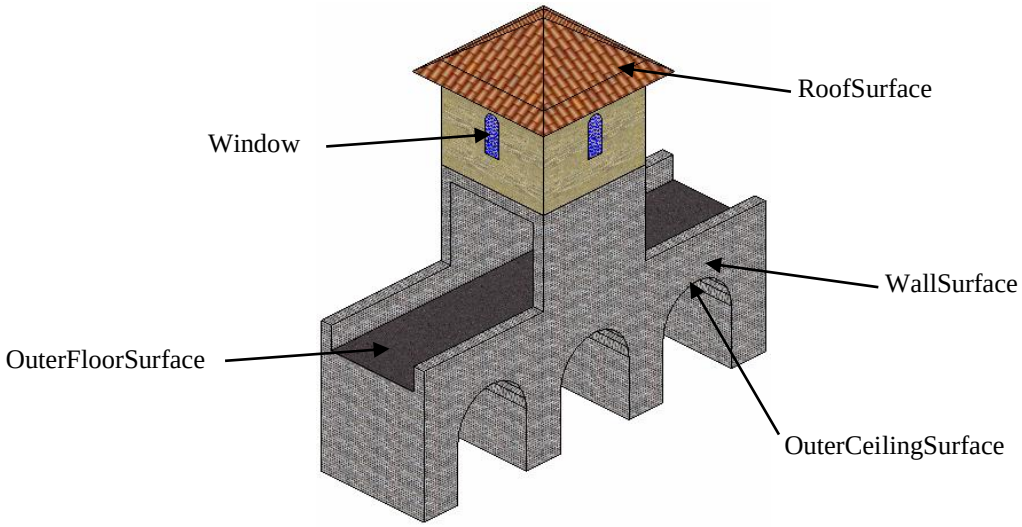

Fig. 50: Different BoundarySurfaces of a bridge.

GroundSurfaceType, GroundSurface

```
<xs:complexType name="GroundSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="GroundSurface" type="GroundSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfGroundSurface" type="xs:anyType" abstract="true"/>
```

The ground plate of a bridge or bridge part is modelled by the class *GroundSurface*. The polygon defining the ground plate is congruent with the bridge's footprint. However, the surface normal of the ground plate is pointing downwards.

OuterCeilingSurfaceType, OuterCeilingSurface

```
<xs:complexType name="OuterCeilingSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfOuterCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="OuterCeilingSurface" type="OuterCeilingSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfOuterCeilingSurface" type="xs:anyType" abstract="true"/>
```

A mostly horizontal surface belonging to the outer bridge shell and having the orientation pointing downwards can be modeled as an *OuterCeilingSurface*.

WallSurfaceType, WallSurface

```
<xs:complexType name="WallSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWallSurface" minOccurs="0" maxOccurs="unbounded"/>
```



```
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="WallSurface" type="WallSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfWallSurface" type="xs:anyType" abstract="true"/>
```

All parts of the bridge facade belonging to the outer bridge shell can be modelled by the class *WallSurface*.

OuterFloorSurfaceType, OuterFloorSurface

```
<xs:complexType name="OuterFloorSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfOuterFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="OuterFloorSurface" type="OuterFloorSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfOuterFloorSurface" type="xs:anyType" abstract="true"/>
```

A mostly horizontal surface belonging to the outer bridge shell and with the orientation pointing upwards can be modeled as an *OuterFloorSurface*.

RoofSurfaceType, RoofSurface

```
<xs:complexType name="RoofSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfRoofSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="RoofSurface" type="RoofSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfRoofSurface" type="xs:anyType" abstract="true"/>
```

The major roof parts of a bridge or bridge part are expressed by the class *RoofSurface*.

ClosureSurfaceType, ClosureSurface

```
<xs:complexType name="ClosureSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="ClosureSurface" type="ClosureSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfClosureSurface" type="xs:anyType" abstract="true"/>
```

An opening in a bridge not filled by a door or window can be sealed by a virtual surface called *ClosureSurface* (cf. chapter 6.4). Hence, bridge with open sides can be virtually closed in order to be able to compute their volume. *ClosureSurfaces* are also used in the interior bridge model. If two rooms with a different are directly

connected without a separating door, a *ClosureSurface* should be used to separate or connect the volumes of both rooms.

FloorSurfaceType, FloorSurface

```
<xs:complexType name="FloorSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="FloorSurface" type="FloorSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfFloorSurface" type="xs:anyType" abstract="true"/>
```

The class *FloorSurface* must only be used in the LOD4 interior bridge model for modelling the floor of a bridge room.

InteriorWallSurfaceType, InteriorWallSurface

```
<xs:complexType name="InteriorWallSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfInteriorWallSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="InteriorWallSurface" type="InteriorWallSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfInteriorWallSurface" type="xs:anyType" abstract="true"/>
```

The class *InteriorWallSurface* must only be used in the LOD4 interior bridge model for modelling the visible surfaces of the bridge room walls.

CeilingSurfaceType, CeilingSurface

```
<xs:complexType name="CeilingSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="CeilingSurface" type="CeilingSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfCeilingSurface" type="xs:anyType" abstract="true"/>
```

The class *CeilingSurface* must only be used in the LOD4 interior bridge model for modelling the ceiling of a bridge room.

10.5.4 Openings

AbstractOpeningType, _Opening

```
<xs:complexType name="AbstractOpeningType" abstract="true">
  <xs:complexContent>
```



```
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfOpening" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_Opening" type="AbstractOpeningType" abstract="true" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfOpening" type="xs:anyType" abstract="true"/>
```

The class *_Opening* is the abstract base class for semantically describing openings like doors or windows in outer or inner boundary surfaces like walls and roofs. Openings only exist in models of LOD3 or LOD4. Each *_Opening* is associated with a *gml:MultiSurface* geometry. Alternatively, the geometry may be given as *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype opening is stored only once in a local coordinate system and referenced by other opening features (see chapter 8.2).

WindowType, Window

```
<xs:complexType name="WindowType">
  <xs:complexContent>
    <xs:extension base="AbstractOpeningType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWindow" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Window" type="WindowType" substitutionGroup="_Opening"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfWindow" type="xs:anyType" abstract="true"/>
```

The class *Window* is used for modelling windows in the exterior shell of a bridge, or hatches between adjacent rooms. The formal difference between the classes *Window* and *Door* is that – in normal cases – *Windows* are not specifically intended for the transit of people or vehicles.

DoorType, Door

```
<xs:complexType name="DoorType">
  <xs:complexContent>
    <xs:extension base="AbstractOpeningType">
      <xs:sequence>
        <xs:element name="address" type="core:AddressPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfDoor" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Door" type="DoorType" substitutionGroup="_Opening"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfDoor" type="xs:anyType" abstract="true"/>
```

The class *Door* is used for modelling doors in the exterior shell of a bridge, or between adjacent rooms. Doors can be used by people to enter or leave a bridge or room. In contrast to a *ClosureSurface* a door may be closed, blocking the transit of people. A *Door* may be assigned zero or more addresses. The corresponding *AddressPropertyType* is defined within the CityGML core module (cf. chapter 10.1.4).

10.5.5 Bridge interior

The classes *BridgeRoom*, *IntBridgeInstallation* and *BridgeFurniture* allow for the representation of the bridge interior. They are designed in analogy to the classes *Room*, *IntBuildingInstallation* and *BuildingFurniture* of the building module and share the same meaning. The bridge interior can only be modeled in LOD4.

BridgeRoomType, BridgeRoom

```
<xs:complexType name="BridgeRoomType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="interiorFurniture" type="InteriorFurniturePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="bridgeRoomInstallation" type="IntBridgeInstallationPropertyType" minOccurs="0"
                    maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfBridgeRoom" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="BridgeRoom" type="BridgeRoomType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBridgeRoom" type="xs:anyType" abstract="true"/>
```

A *BridgeRoom* is a semantic object for modelling the free space inside a bridge and should be uniquely related to exactly one bridge or bridge part object. It should be closed (if necessary by using *ClosureSurfaces*) and the geometry normally will be described by a solid (*lod4Solid*). However, if the topological correctness of the boundary cannot be guaranteed, the geometry can alternatively be given as a *MultiSurface* (*lod4MultiSurface*). The surface normals of the outer shell of a GML solid must point outwards. This is important to consider when *BridgeRoom* surfaces should be assigned *Appearances*. In this case, textures and colors must be placed on the backside of the corresponding surfaces in order to be visible from the inside of the room.

In addition to the geometrical representation, different parts of the visible surface of a room can be modelled by specialised *BoundarySurfaces* (*FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface;* cf. chapter 10.5.3).

BridgeFurnitureType, BridgeFurniture

```
<xs:complexType name="BridgeFurnitureType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfBridgeFurniture" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="BridgeFurniture" type="BridgeFurnitureType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBridgeFurniture" type="xs:anyType" abstract="true"/>
```

*BridgeRoom*s may have *BridgeFurnitures* and *IntBridgeInstallations*. A *BridgeFurniture* is a movable part of a room, such as a chair or furniture. A *BridgeFurniture* object should be uniquely related to exactly one room

object. Its geometry may be represented by an explicit geometry or an *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype bridge furniture is stored only once in a local coordinate system and referenced by other bridge furniture features (see chapter 8.2).

IntBridgeInstallationType, IntBridgeInstallation

```
<xs:complexType name="IntBridgeInstallationType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfIntBridgeInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="IntBridgeInstallation" type="IntBridgeInstallationType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfIntBridgeInstallation" type="xs:anyType" abstract="true"/>
```

An *IntBridgeInstallation* is an object inside a bridge with a specialised function or semantic meaning. In contrast to *BridgeFurniture*, *IntBridgeInstallations* are permanently attached to the bridge structure and cannot be moved. Examples for *IntBridgeInstallations* are stairways, railings and heaters. Objects of the class *IntBridgeInstallation* can either be associated with a room (class *BridgeRoom*), or with the complete bridge / bridge part (class *_AbstractBridge*, cf. chapter 10.5.1). However, they should be uniquely related to exactly one room or one bridge / bridge part object. An *IntBridgeInstallation* optionally has attributes *class*, *function* and *usage*. The attribute *class*, which can only occur once, represents a general classification of the internal bridge component. With the attributes *function* and *usage*, nominal and real functions of a bridge installation can be described. For all three attributes the list of feasible values can be specified in a code list. For the geometrical representation of an *IntBridgeInstallation*, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used. Alternatively, the geometry may be given as *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype interior bridge installation is stored only once in a local coordinate system and referenced by other interior bridge installation features (see chapter 8.2). The visible surfaces of an interior bridge installation can be semantically classified using the concept of boundary surfaces (cf. 10.5.3).

10.5.6 Examples

The bridge of Rees crossing the Rhine in Germany has three bridge parts which are separated by pylons. Fig. 51 (left) depicts the Rees bridge model containing one *Bridge* feature which consists of three *BridgePart* features. The pylons, which are structurally essential, are represented by *BridgeConstructionElements*. On the top of the pylons, four lamps are located which are modeled as *BridgeInstallation* features (cf. right part of Fig. 51).

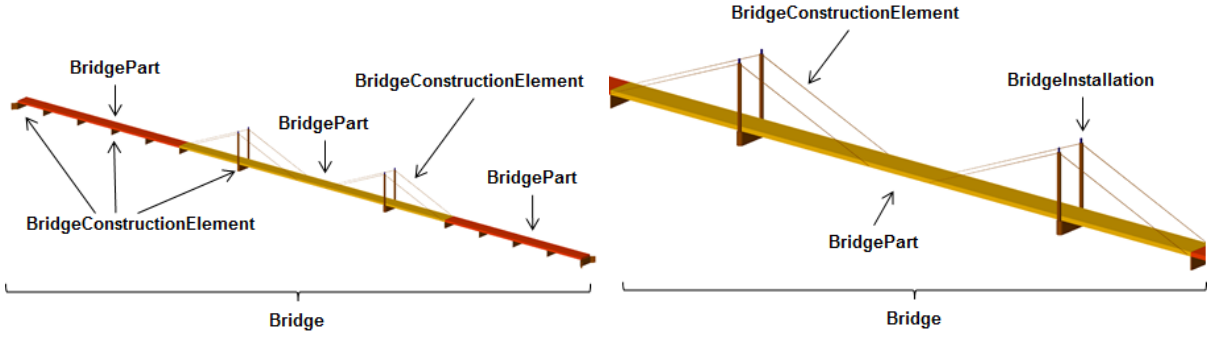

Fig. 51: The bridge of Rees, consisting of a *Bridge* feature and three *BridgePart* features (left). The bridge contains *BridgeConstructionElement* and *BridgeInstallation* features (right).

In the following Fig. 52, the main part of the bridge of Rees is shown as photograph on the left side (source: Harald Halfpapp), and the corresponding part of the LOD2 bridge model is depicted on the the right side (source: District of Recklinghausen / KIT).

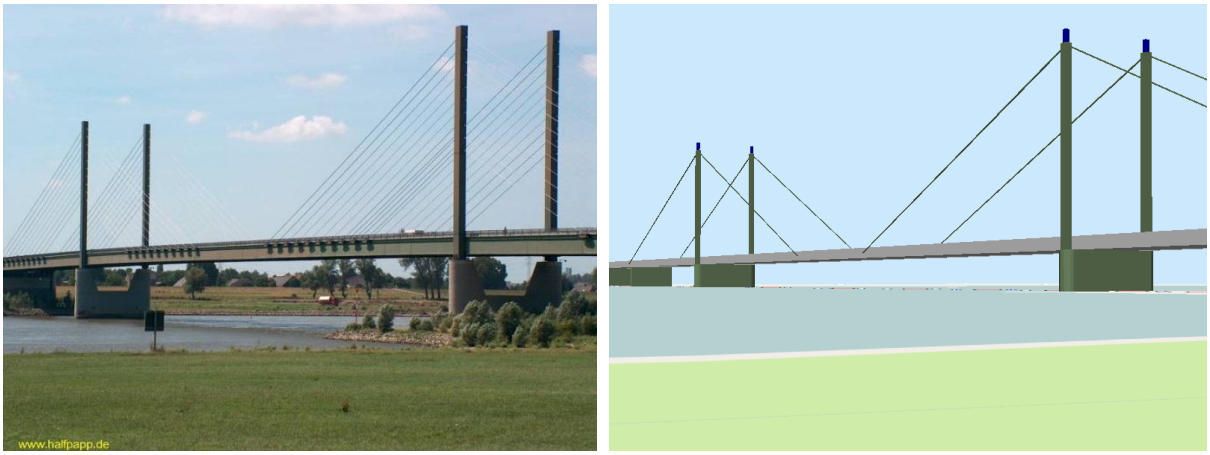

Fig. 52: The bridge of Rees (left photo (source: Harald Halfpapp); right LOD2 model (source: District of Recklinghausen / KIT)).

There are two bridges crossing the river Rhine at Karlsruhe, Germany. The first one is a two track railway bridge constructed as a truss bridge (cf. Fig. 53 front). The second one is a four lane highway bridge constructed as a cable-stayed bridge (cf. Fig. 53 background).

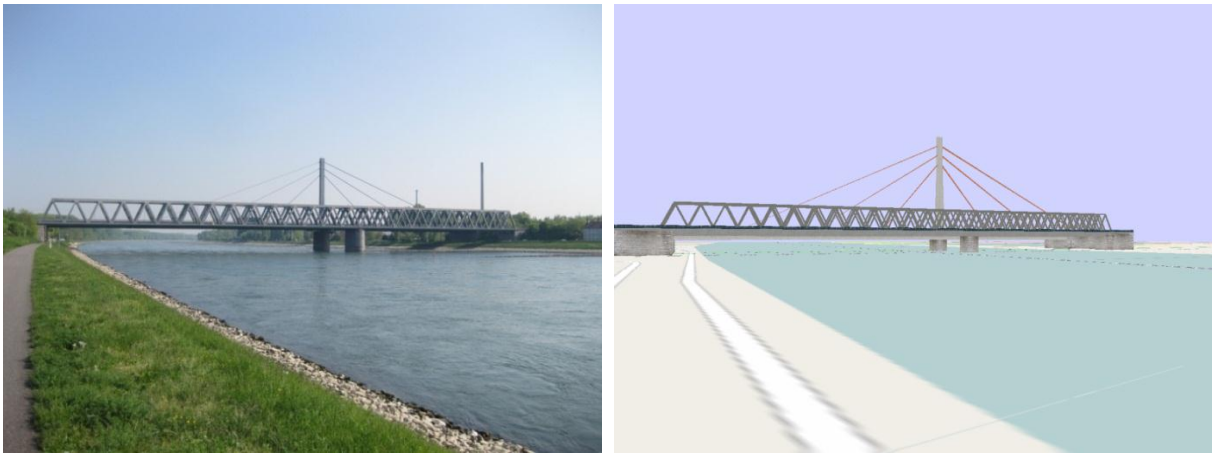

Fig. 53: Bridge over the river Rhine at Karlsruhe (left a photo, right the 3D CityGML model) (source: Karlsruhe Institute of Technology (KIT), courtesy of City of Karlsruhe).

In CityGML both bridges are modeled as single *Bridge* object with *BridgeConstructionElements* and *BridgeInstallations*. The construction elements of the cable stayed bridge are the footings on both river sides and in the middle of the river, as well as the cables and the pylon. The construction elements of the truss bridge are the footings and the truss itself. Both bridges have several railings which are modeled as *BridgeInstallation*.

The bridge “Oberbaumbrücke” shown in Fig. 54 is located in the centre of Berlin crossing the river Spree and serves as example for bridges having interior rooms. The real-world bridge is depicted in the left part of Fig. 54, whereas the corresponding CityGML model is shown on the right. The outer geometry of the bridge is modeled as *gml:MultiSurface* element (*lod4MultiSurface* property) and is assigned photorealistic textures. Additionally, the interior rooms located in both bridge towers are represented as *BridgeRoom* objects with solid geometries (*gml:Solid* assigned through the *lod4Solid* property). Due to its high geometric accuracy and the representation of the interior structures of both bridge towers, the model is classified as LOD4.

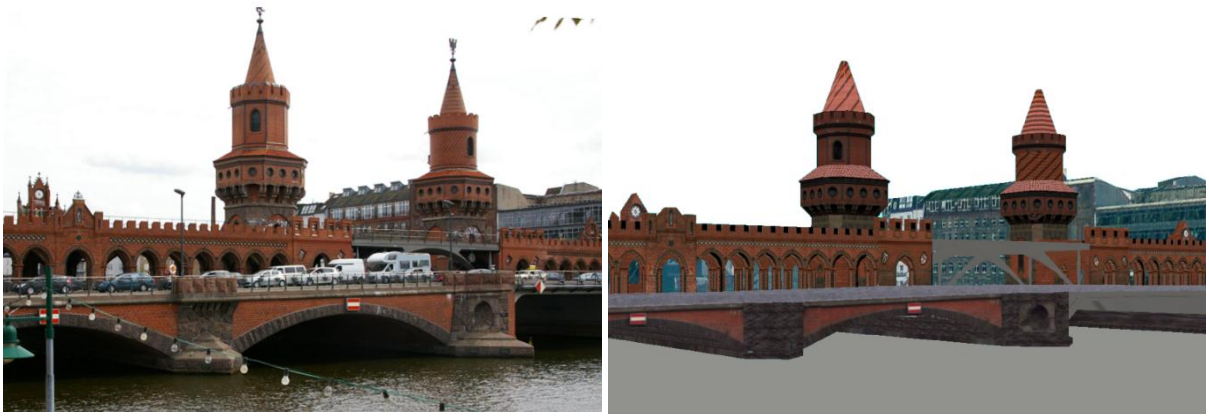

Fig. 54: The bridge“Oberbaumbrücke” in Berlin represented as bridge model in LOD4 (left a photo, right the 3D CityGML model) (source: Berlin Senate of Business, Technology and Women; Business Location Center, Berlin; Technische Universität Berlin; Karlsruhe Institute of Technology (KIT)).

10.5.7 Code lists

The attributes *class*, *function*, and *usage* of the features *_AbstractBridge*, *BridgeConstructionElement*, *BridgeInstallation*, *BridgeRoom*, *BridgeFurniture* and *IntBridgeInstallation* are specified as *gml:CodeType*. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.3.

10.5.8 Conformance requirements

Base requirements

1. If a bridge only consists of one (homogeneous) part, it shall be represented by the element *Bridge*. However, if a bridge is composed of individual structural segments, it shall be modelled as a *Bridge* element having one or more additional *BridgePart* elements. Only the geometry and non-spatial properties of the main part of the bridge should be represented within the aggregating *Bridge* element.

Usage restriction of bridge model components according to different LODs

2. The *lodXSolid* and *lodXMultiSurface*, X ∈ [1..4], properties (*gml:SolidPropertyType* resp. *gml:MultiSurfacePropertyType*) of *_AbstractBridge* may be used to geometrically represent the exterior shell of a bridge (as volume or surface model) within each LOD. For LOD1, either *lod1Solid* or *lod1MultiSurface* must be used, but not both. Starting from LOD2, both properties may be modelled individually and complementary.
3. Starting from LOD2, the exterior shell of an *_AbstractBridge* may be semantically decomposed into *_BoundarySurface* elements using the *boundedBy* property (type: *BoundarySurfacePropertyType*) of *_AbstractBridge*. Only *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface* and *ClosureSurface* as subclasses of *_BoundarySurface* are allowed. The *boundedBy* property (not to be confused with the *gml:boundedBy* property) shall not be used if the bridge is only represented in LOD1.

   If the exterior shell is represented by *_BoundarySurface* elements, an additional geometric representation as volume or surface model using the *lodXSolid* and *lodXMultiSurface*, X ∈ [2..4], properties shall not explicitly define the geometry, but has to reference the according components of the *gml:MultiSurface* element of *_BoundarySurface* within each LOD using the XLink concept of GML 3.1.1.
4. Starting from LOD2, curve parts of the bridge shell may be represented using the *lodXMultiCurve*, X ∈ [2..4], property of *_AbstractBridge*. This property shall not be used if the bridge is only represented in LOD1.
5. Starting from LOD1, the *outerBridgeConstruction* property (type: *BridgeConstructionElementPropertyType*) of *_AbstractBridge* may be used to model *BridgeConstructionElement* elements. *BridgeCon-*

structionElement elements shall only be used to represent outer characteristics of a bridge which do not have the significance of bridge parts and are essential from a structural point of view.

6. Starting from LOD2, the geometry of *BridgeConstructionElement* elements may be semantically classified by *_BoundarySurface* elements using the *boundedBy* property (type: *BoundarySurfacePropertyType*) of *BridgeConstructionElement*. The *boundedBy* property (not to be confused with the *gml:boundedBy* property) shall not be used if the bridge construction element is only represented in LOD1.
7. Starting from LOD2, the *outerBridgeInstallation* property (type: *BridgeInstallationPropertyType*) of *_AbstractBridge* may be used to model *BridgeInstallation* elements. *BridgeInstallation* elements shall only be used to represent outer characteristics of a bridge which do not have the significance of bridge parts and are not essential from a structural point of view.
8. Starting from LOD2, the geometry of *BridgeInstallation* elements may be semantically classified by *_BoundarySurface* elements using the *boundedBy* property (type: *BoundarySurfacePropertyType*) of *BridgeInstallation*. Only *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface* and *ClosureSurface* as subclasses of *_BoundarySurface* are allowed.
9. Starting from LOD3, openings of *_BoundarySurface* elements may be modelled using the *opening* property (type: *OpeningPropertyType*) of *_BoundarySurface*. This property shall not be used for *_BoundarySurface* elements only represented in LOD2. Accordingly, the surface geometry representing a *_BoundarySurface* in LOD2 must be simply connected.

   The *opening* property of *_BoundarySurface* may contain or reference *_Opening* elements. If the geometric location of an *_Opening* element topologically lies within a surface component of the *_BoundarySurface*, the opening must also be represented as inner hole of that surface. The embrasure surface of an *_Opening* element shall belong to the relevant adjacent *_BoundarySurface*.
10. Starting from LOD4, the *interiorBridgeRoom* property (type: *InteriorBridgeRoomPropertyType*) of *_AbstractBridge* may be used to semantically model the free space inside the bridge by *BridgeRoom* elements. This property shall not be used if the bridge is only represented in LOD 1 – 3. The *BridgeRoom* element may be geometrically represented as a surface or volume model, using its *lod4Solid* or *lod4MultiSurface* property (*gml:SolidPropertyType* resp. *gml:MultiSurfacePropertyType*).

    In addition, different parts of the visible surface of a bridge room may be modelled by thematic *_BoundarySurface* elements. Only *FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface* as subclasses of *_BoundarySurface* are allowed. If the visible surface of a room is represented by *_BoundarySurface* elements, an additional geometric representation as volume or surface model using the *lod4Solid* and *lod4MultiSurface* property shall not explicitly define the geometry, but has to reference the according components of the *gml:MultiSurface* element of *_BoundarySurface* using the XLink concept of GML 3.1.1.
11. Starting from LOD4, the *interiorBridgeInstallation* property (type: *IntBridgeInstallationPropertyType*) of *_AbstractBridge* may be used to represent immovable objects inside the bridge that are permamently attached to the bridge structure. The *interiorBridgeInstallation* property shall not be used if the bridge is only represented in LOD 1 – 3. Furthermore, the *interiorBridgeInstallation* property shall only be used if the object cannot be associated with a *BridgeRoom* element. In the latter case, the *bridgeRoomInstallation* property (type: *IntBridgeInstallationPropertyType*) of the corresponding *BridgeRoom* element shall be used to represent the object.
12. Starting from LOD4, the geometry of *IntBridgeInstallation* elements may be semantically classified by *_BoundarySurface* elements using the *boundedBy* property (type: *BoundarySurfacePropertyType*) of *IntBridgeInstallation*. Only *FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface* as subclasses of *_BoundarySurface* are allowed.

Referential integrity

13. The *boundedBy* property (type: *BoundarySurfacePropertyType*) of the element *_AbstractBridge* may contain a *_BoundarySurface* element inline or an XLink reference to a remote *_BoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_BoundarySurface* element (where remote *_BoundarySurface* ele-

ments are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

Only *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface* and *ClosureSurface* elements are allowed to be encapsulated or referenced by the boundedBy property of *_AbstractBridge*.

14. The *outerBridgeConstruction* property (type: *BridgeConstructionElementPropertyType*) of the element *_AbstractBridge* may contain a *BridgeConstructionElement* element inline or an XLink reference to a remote *BridgeConstructionElement* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *outerBridgeConstruction* property may only point to a remote *BridgeConstructionElement* element (where remote *BridgeConstructionElement* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
15. The *outerBridgeInstallation* property (type: *BridgeInstallationPropertyType*) of the element *_AbstractBridge* may contain a *BridgeInstallation* element inline or an XLink reference to a remote BridgeInstallation element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *outerBridgeInstallation* property may only point to a remote *BridgeInstallation* element (where remote *BridgeInstallation* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
16. The *interiorBridgeInstallation* property (type: *IntBridgeInstallationPropertyType*) of the element *_AbstractBridge* may contain an *IntBridgeInstallation* element inline or an XLink reference to a remote *IntBridgeInstallation* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *interiorBridgeInstallation* property may only point to a remote *IntBridgeInstallation* element (where remote *IntBridgeInstallation* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
17. The *interiorBridgeRoom* property (type: *InteriorBridgeRoomPropertyType*) of the element *_AbstractBridge* may contain a *BridgeRoom* element inline or an XLink reference to a remote *BridgeRoom* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *interiorBridgeRoom* property may only point to a remote *BridgeRoom* element (where remote *BridgeRoom* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
18. The *consistsOfBridgePart* property (type: *BridgePartPropertyType*) of the element *_AbstractBridge* may contain a *BridgePart* element inline or an XLink reference to a remote *BridgePart* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *consistsOfBridgePart* property may only point to a remote *BridgePart* element (where remote *BridgePart* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
19. The *address* property (type: *core:AddressPropertyType*) of the element *_AbstractBridge* may contain an *core:Address* element inline or an XLink reference to a remote *core:Address* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the address property may only point to a remote *core:Address* element (where remote *core:Address* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
20. The *opening* property (type: *OpeningPropertyType*) of the element *_BoundarySurface* may contain an *_Opening* element inline or an XLink reference to a remote *_Opening* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the opening property may only point to a remote *_Opening* element (where remote *_Opening* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
21. The *address* property (type: *core:AddressPropertyType*) of the element *Door* may contain an *core:Address* element inline or an XLink reference to a remote *core:Address* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the address property may only point

to a remote *core:Address* element (where remote *core:Address* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

22. The *boundedBy* property (type: *BoundarySurfacePropertyType*) of the element *BridgeConstructionElement* may contain a *_BoundarySurface* element inline or an XLink reference to a remote *_BoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_BoundarySurface* element (where remote *_BoundarySurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
23. The *boundedBy* property (type: *BoundarySurfacePropertyType*) of the element *BridgeInstallation* may contain a *_BoundarySurface* element inline or an XLink reference to a remote *_BoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_BoundarySurface* element (where remote *_BoundarySurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

    Only *RoofSurface*, *WallSurface*, *GroundSurface*, *OuterCeilingSurface*, *OuterFloorSurface* and *ClosureSurface* elements are allowed to be encapsulated or referenced by the *boundedBy* property of *BridgeInstallation*.
24. The *boundedBy* property (type: *BoundarySurfacePropertyType*) of the element *IntBridgeInstallation* may contain a *_BoundarySurface* element inline or an XLink reference to a remote *_BoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_BoundarySurface* element (where remote *_BoundarySurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

    Only *FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface* elements are allowed to be encapsulated or referenced by the *boundedBy* property of *IntBridgeInstallation*.
25. The *boundedBy* property (type: *BoundarySurfacePropertyType*) of the element *BridgeRoom* may contain a *_BoundarySurface* element inline or an XLink reference to a remote *_BoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_BoundarySurface* element (where remote *_BoundarySurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

    Only *FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, and *ClosureSurface* elements are allowed to be encapsulated or referenced by the *boundedBy* property of *BridgeRoom*.
26. The *interiorFurniture* property (type: *InteriorFurniturePropertyType*) of the element *BridgeRoom* may contain an *BridgeFurniture* element inline or an XLink reference to a remote *BridgeFurniture* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *interiorFurniture* property may only point to a remote *BridgeFurniture* element (where remote *BridgeFurniture* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
27. The *bridgeRoomInstallation* property (type: *IntBridgeInstallationPropertyType*) of the element *BridgeRoom* may contain an *IntBridgeInstallation* element inline or an XLink reference to a remote *IntBridgeInstallation* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *bridgeRoomInstallation* property may only point to a remote *IntBridgeInstallation* element (where remote *IntBridgeInstallation* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
28. The *lodXImplicitRepresentation*, X ∈ [1..4], property (type: *core:ImplicitRepresentationPropertyType*) of the element *BridgeConstructionElement* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lodXImplicitRepresentation*, X ∈ [1..4], property may only

point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

29. The *lodXImplicitRepresentation*, X ∈ [2..4], property (type: *core:ImplicitRepresentationPropertyType*) of the element *BridgeInstallation* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lodXImplicitRepresentation*, X ∈ [2..4], property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
30. The *lodXImplicitRepresentation*, X ∈ [3..4], property (type: *core:ImplicitRepresentationPropertyType*) of the element *_Opening* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lodXImplicitRepresentation*, X ∈ [3..4], property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
31. The *lod4ImplicitRepresentation* property (type: *core:ImplicitRepresentationPropertyType*) of the element *IntBridgeInstallation* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lod4ImplicitRepresentation* property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
32. The *lod4ImplicitRepresentation* property (type: *core:ImplicitRepresentationPropertyType*) of the element *BridgeFurniture* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lod4ImplicitRepresentation* property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

10.6 Water bodies

Waters have always played an important role in urbanisation processes and cities were built preferably at rivers and places where landfall seemed to be easy. Obviously, water is essential for human alimentation and sanitation. Water bodies present the most economical way of transportation and are barriers at the same time, that avoid instant access to other locations. Bridging waterways caused the first efforts of construction and resulted in high-tech bridges of today. The landscapes of many cities are dominated by water, which directly relates to 3D city models. Furthermore, water bodies are important for urban life as subject of recreation and possible hazards as e.g. floods.

The distinct character of water bodies compared with the permanence of buildings, roadways, and terrain is considered in this thematic model. Water bodies are dynamic surfaces. Tides occur regularly, but irregular events predominate with respect to natural forces, for example flood events. The visible water surface changes in height and its covered area with the necessity to model its semantics and geometry distinct from adjacent objects like terrain or buildings.

This first modelling approach of water bodies fulfils the requirements of 3D city models. It does not inherit any hydrological or other dynamic aspects. In these terms it does not claim to be complete. However, the semantic and geometric description given here allows further enhancements of dynamics and conceptually different descriptions. The water bodies model of CityGML is embraced by the thematic extension module *WaterBody* (cf. chapter 7).

The water bodies model represents the thematic aspects and three-dimensional geometry of rivers, canals, lakes, and basins. In the LOD 2-4 water bodies are bounded by distinct thematic surfaces. These surfaces are the obligatory *WaterSurface*, defined as the boundary between water and air, the optional *WaterGroundSurface*, defined as the boundary between water and underground (e.g. DTM or floor of a 3D basin object), and zero or more *WaterClosureSurfaces*, defined as virtual boundaries between different water bodies or between water and the end of a modelled region (see Fig. 55). A dynamic element may be the *WaterSurface* to represent temporarily changing situations of tidal flats.

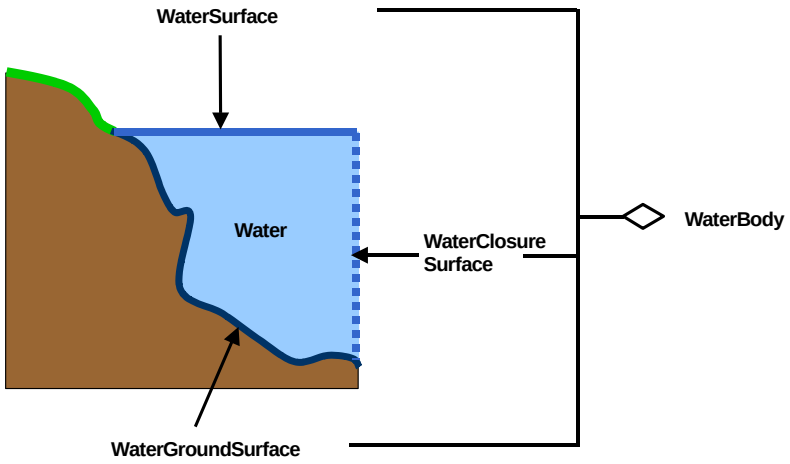

Fig. 55: Illustration of a water body defined in CityGML (graphic: IGG Uni Bonn).

The UML diagram of the water body model is depicted in Fig. 56, for the XML schema definition see below and annex A.13. Each *WaterBody* object may have the attributes *class*, *function* and *usage* whose possible values can be enumerated in code lists (cf. chapter 10.6.3 and annex C.9). The attribute *class* defines the classification of the object, e.g. lake, river, or fountain and can occur only once. The attribute *function* contains the purpose of the object like, for example national waterway or public swimming, while the attribute *usage* defines the actual usages, e.g. whether the water body is navigable. The latter two attributes can occur multiple times.

WaterBody is a subclass of *_WaterObject* and transitively of the root class *_CityObject*. The class *_WaterObject* may be differentiated in further subclasses of water objects in the future. The geometrical representation of the *WaterBody* varies through the different levels of detail. Since *WaterBody* is a subclass of *_CityObject* and hence a feature, it inherits the attribute *gml:name*. The *WaterBody* can be differentiated semantically by the class *_WaterBoundarySurface*. A *_WaterBoundarySurface* is a part of the water body's exterior shell with a special

function like *WaterSurface*, *WaterGroundSurface* or *WaterClosureSurface*. As with any *_CityObject*, *WaterBody* objects as well as *WaterSurface*, *WaterGroundSurface*, and *WaterClosureSurface* may be assigned *ExternalReferences* (cf. chapter 6.7) and may be augmented by *generic attributes* using CityGML's *Generics* module (cf. chapter 10.12).

The optional attribute *waterLevel* of a *WaterSurface* can be used to describe the water level, for which the given 3D surface geometry was acquired. This is especially important when the water body is influenced by the tide. The allowed values can be defined in a corresponding code list.

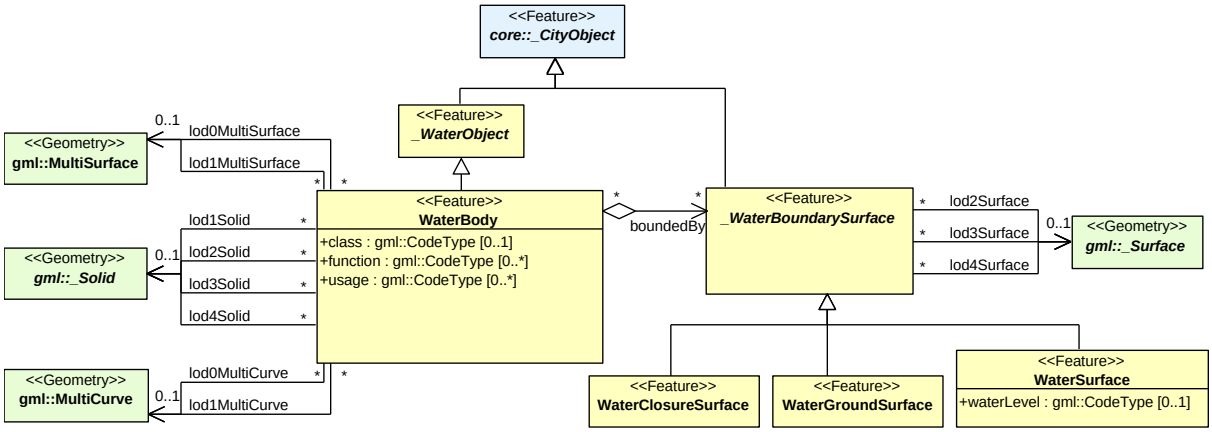

Fig. 56: UML diagram of the water body model in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *WaterBody* module.

Both LOD0 and LOD1 represent a low level of illustration and high grade of generalisation. Here the rivers are modelled as *MultiCurve* geometry and brooks are omitted. Seas, oceans and lakes with significant extent are represented as a *MultiSurface* (Fig. 56). Every *WaterBody* may be assigned a combination of geometries of different types. Linear water bodies are represented as a network of 3D curves. Each curve is composed of straight line segments, where the line orientation denotes the flow direction (water flows from the first point of a curve, e.g. a *gml:LineString*, to the last). Areal objects like lakes or seas are represented by 3D surface geometries of the water surface.

Starting from LOD1 water bodies may also be modelled as water filled volumes represented by *Solids*. If a water body is represented by a *gml:Solid* in LOD2 or higher, the surface geometries of the corresponding thematic *WaterClosureSurface*, *WaterGroundSurface*, and *WaterSurface* objects must coincide with the exterior shell of the *gml:Solid*. This can be ensured, if for each LOD *X* the respective *lodXSolid* representation (where *X* is between 2 and 4) does not redundantly define the geometry, but instead references the corresponding polygons (using GML3's XLink mechanism) of the *lodXSurface* elements (where *X* is between 2 and 4) of *WaterClosureSurface*, *WaterGroundSurface*, and *WaterSurface*.

LOD2 to LOD4 demand a higher grade of detail and therefore any *WaterBody* can be outlined by thematic surfaces or a solid composed of the surrounding thematic surfaces.

Every object of the class *WaterSurface*, *WaterClosureSurface*, and *WaterGroundSurface* must have at least one associated surface geometry. This means, that every *WaterSurface*, *WaterClosureSurface*, and *WaterGroundSurface* feature within a CityGML instance document must contain at least one of the following properties: *lod2Surface*, *lod3Surface*, *lod4Surface*.

The water body model implicitly includes the concept of *TerrainIntersectionCurves* (TIC), e.g. to specify the exact intersection of the DTM with the 3D geometry of a *WaterBody* or to adjust a *WaterBody* or *WaterSurface* to the surrounding DTM (see chapter 6.5). The rings defining the *WaterSurface* polygons implicitly delineate the intersection of the water body with the terrain or basin.

XML namespace

The XML namespace of the CityGML *WaterBody* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/waterbody/2.0. Within the XML Schema definition of the *WaterBody* module, this URI is also used to identify the default namespace.

10.6.1 Water body

AbstractWaterObjectType, _WaterObject

```
<xs:complexType name="AbstractWaterObjectType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWaterObject" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_WaterObject" type="AbstractWaterObjectType" abstract="true" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfWaterObject" type="xs:anyType" abstract="true"/>
```

WaterBodyType, WaterBody

```
<xs:complexType name="WaterBodyType">
  <xs:complexContent>
    <xs:extension base="AbstractWaterObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod0MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod0MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod1MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod1Solid" type="gml:SolidPropertyType" minOccurs="0"/>
        <xs:element name="lod2Solid" type="gml:SolidPropertyType" minOccurs="0"/>
        <xs:element name="lod3Solid" type="gml:SolidPropertyType" minOccurs="0"/>
        <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundedByWaterSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfWaterBody" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="WaterBody" type="WaterBodyType" substitutionGroup="_WaterObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfWaterBody" type="xs:anyType" abstract="true"/>
```

10.6.2 Boundary surfaces

With respect to different functions and characteristics three boundary classes for water are defined to build a solid or composite surface geometry (Fig. 55).

1. Boundary class “Air to Water”. The *WaterSurface* is mandatory to the model and usually is registered using photogrammetric analysis or mapping exploration. The representation may vary due to tidal flats or changing water levels, which can be reflected by including different static water surfaces having different *waterLevels* (*gml:CodeType)*, as for example highest flooding event, mean sea level, or minimum water level. This offers the opportunity to describe significant water surfaces due to levels that are important for certain representations e.g. in tidal zones.
2. Boundary class “Water to Ground”. The *WaterGroundSurface* may be known by sonar exploration or other depth measurements. Also part of the ground surface is the boundary “Water to Construction”. The ground surface might be identical to the underwater terrain model, but also describes the contour to other underwater objects. The usefulness of this concept arises from the existence of water defence constructions like sluices, sills, flood barrage or tidal power stations. The use of *WaterGroundSurface* as boundary layer to man-made constructions is relevant in urban situations, where such objects may enclose the modeled water body completely, for example fountains and swimming pools. The *WaterSurface* objects together with the *WaterGroundSurface* objects enclose the *WaterBody* as a volume.

3. Boundary class “Water to Water”. The *WaterClosureSurface* is an optional feature that comes in use when the union of the *WaterSurfaces* and *WaterGroundSurfaces* of a water body does not define a closed volume. The *WaterClosureSurface* is then used to complete the enclosure of water volumes and to separate water volumes from those where only the surface is known. This might occur, where the cross section and ground surface of rivers is partly available during its course.

_WaterBoundarySurfaces should only be included as parts of corresponding *WaterBody* objects and should not be used as stand-alone objects within a CityGML model.

AbstractWaterBoundarySurfaceType, _WaterBoundarySurface

```
<xs:complexType name="AbstractWaterBoundarySurfaceType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="lod2Surface" type="gml:SurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3Surface" type="gml:SurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4Surface" type="gml:SurfacePropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfWaterBoundarySurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_WaterBoundarySurface" type="AbstractWaterBoundarySurfaceType" abstract="true"
            substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfWaterBoundarySurface" type="xs:anyType" abstract="true"/>
```

WaterSurfaceType, WaterSurface

```
<xs:complexType name="WaterSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractWaterBoundarySurfaceType">
      <xs:sequence>
        <xs:element name="waterLevel" type="gml:CodeType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfWaterSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="WaterSurface" type="WaterSurfaceType" substitutionGroup="_WaterBoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfWaterSurface" type="xs:anyType" abstract="true"/>
```

WaterGroundSurfaceType, WaterGroundSurface

```
<xs:complexType name="WaterGroundSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractWaterBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWaterGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="WaterGroundSurface" type="WaterGroundSurfaceType" substitutionGroup="_WaterBoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfWaterGroundSurface" type="xs:anyType" abstract="true"/>
```

WaterClosureSurfaceType, WaterClosureSurface

```
<xs:complexType name="WaterClosureSurfaceType">
  <xs:complexContent>
```

```
    <xs:extension base="AbstractWaterBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWaterClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="WaterClosureSurface" type="WaterClosureSurfaceType" substitutionGroup="_WaterBoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfWaterClosureSurface" type="xs:anyType" abstract="true"/>
```

10.6.3 Code lists

The attributes *class*, *function*, and *usage* of the feature *WaterBody* as well as the attribute *waterLevel* of the feature *WaterSurface* are specified as *gml:CodeType*. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.9.

10.6.4 Conformance requirements

Base requirements

1. For LOD0 and LOD1, the geometry of a *WaterBody* may be modelled as a linear network using *gml:MultiCurve* geometry elements. In that case, each *gml:MultiCurve* shall be composed of straight line segements, where the line orientation denotes the flow direction. The flow direction is from the first point of a line segment to its last point.
2. Starting from LOD2, the exterior shell of a *WaterBody* may be semantically decomposed into *_WaterBoundarySurface* elements using the *boundedBy* property (type: *BoundedByWaterSurfacePropertyType*) of *WaterBody*.The *boundedBy* property shall not be used if the water body is only represented in lower LODs.

   If the exterior shell is represented by *_WaterBoundarySurface* elements, an additional geometric representation as volume model using the *lodXSolid*, X ∈ [2..4], property of *WaterBody* shall not explicitly define the geometry, but has to reference the according *gml:_Surface* elements of the *_WaterBoundarySurface* objects within each LOD using the XLink concept of GML 3.1.1.
3. Each *_WaterBoundarySurface* element must have at least one associated surface geometry given by the *lodXSurface*, X ∈ [2..4], properties of *_WaterBoundarySurface*.
4. *_WaterBoundarySurface* elements shall only be included as parts of corresponding *WaterBody* elements. They may not be given as stand-alone city objects within a CityGML model.

Referential integrity

5. The *boundedBy* property (type: *BoundedByWaterSurfacePropertyType*) of the element *WaterBody* may contain a *_WaterBoundarySurface* element inline or an XLink reference to a remote *_WaterBoundarySurface* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *boundedBy* property may only point to a remote *_WaterBoundarySurface* element (where remote *_WaterBoundarySurface* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

10.7 Transportation objects

The transportation model of CityGML is a multi-functional, multi-scale model focusing on thematic and functional as well as on geometrical/topological aspects. Transportation features are represented as a linear network in LOD0. Starting from LOD1, all transportation features are geometrically described by 3D surfaces. The areal modelling of transportation features allows for the application of geometric route planning algorithms. This can be useful to determine restrictions and manoeuvres required along a transportation route. This information can also be employed for trajectory planning of mobile robots in the real world or the automatic placement of avatars (virtual people) or vehicle models in 3D visualisations and training simulators. The transportation model of CityGML is provided by the thematic extension module *Transportation* (cf. chapter 7).

The main class is *TransportationComplex*, which represents, for example, a road, a track, a railway, or a square. Fig. 57 illustrates the four different thematic classes.

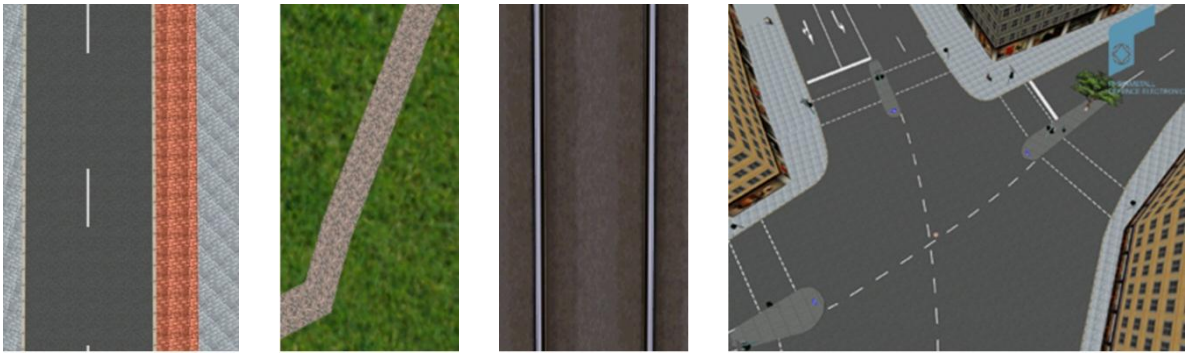

Fig. 57: Representations of *TransportationComplex* (from left to right: examples of road, track, rail, and square) (source: Rheinmetall Defence Electronics).

A *TransportationComplex* is composed of the parts *TrafficArea* and *AuxiliaryTrafficArea*. Fig. 58 depicts an example for a LOD2 *TransportationComplex* configuration within a virtual 3D city model. The *Road* consists of several *TrafficAreas* for the sidewalks, road lanes, parking lots, and of *AuxiliaryTrafficAreas* below the raised flower beds.

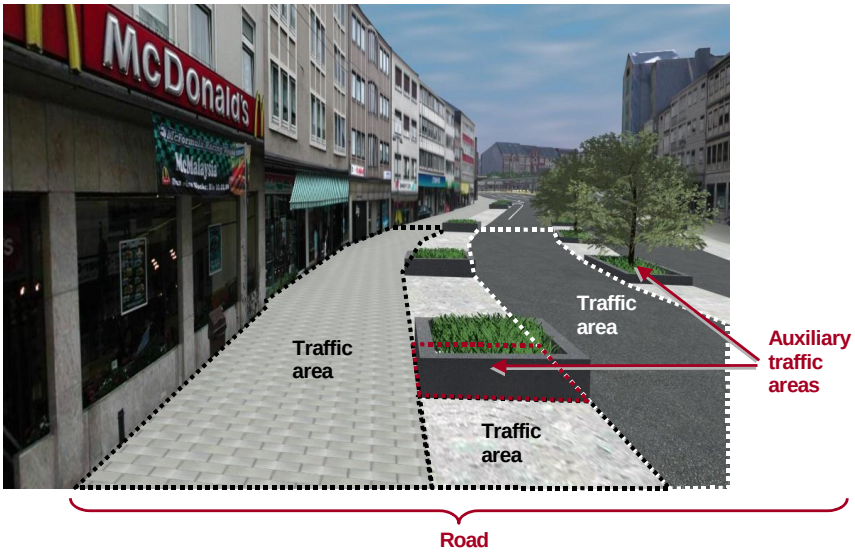

Fig. 58: Example for the representation of a *TransportationComplex* in LOD2 in CityGML: a road, which is the aggregation of *TrafficAreas* and *AuxiliaryTrafficAreas* (source: City of Solingen, IGG Uni Bonn).

Fig. 59 depicts the UML diagram of the transportation model, for the XML schema definition see annex A.10.

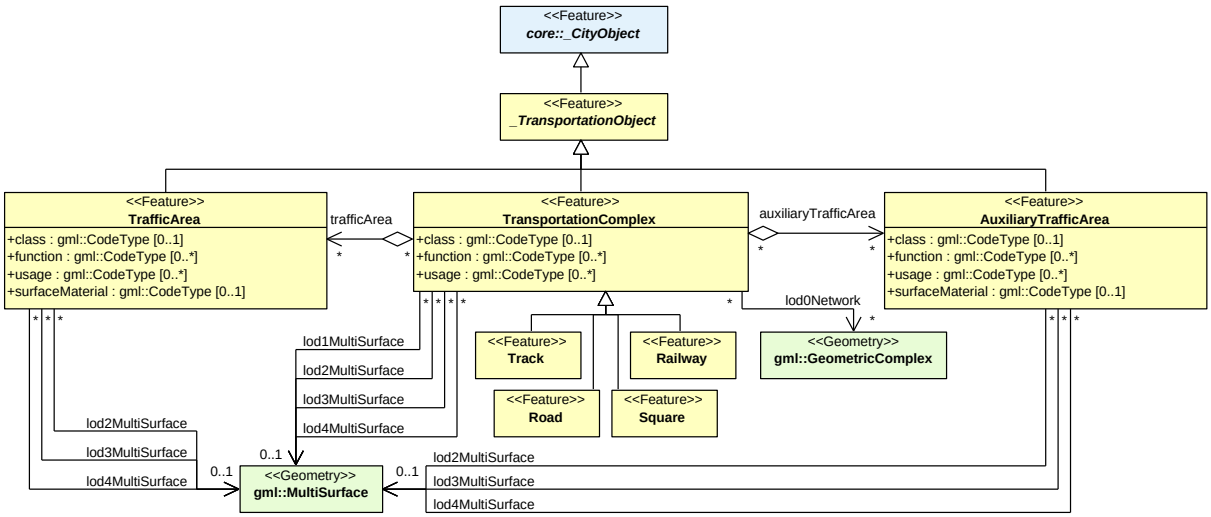

Fig. 59: UML diagram of the transportation model in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Transportation* module.

The road itself is represented as a *TransportationComplex*, which is further subdivided into *TrafficAreas* and *AuxiliaryTrafficAreas*. The *TrafficAreas* are those elements, which are important in terms of traffic usage, like car driving lanes, pedestrian zones and cycle lanes. The *AuxiliaryTrafficAreas* are describing further elements of the road, like kerbstones, middle lanes, and green areas.

TransportationComplex objects can be thematically differentiated using the subclasses *Track, Road, Railway,* and *Square.* Every *TransportationComplex* has the attributes *class*, *function* and *usage* whose possible values can be enumerated in code lists (chapter 10.7.4 and annex C.8). The attribute *class* describes the classification of the object, *function* describes the purpose of the object, for example national motorway, country road, or airport, while the attribute *usage* can be used, if the actual usage differs from the function. The attributes *function* and *usage* can occur multiple times.

In addition both *TrafficArea* and *AuxiliaryTrafficArea* may have the attributes *class*, *function*, *usage*, and *surfaceMaterial*. The attribute *class* describes the classification of the object. For *TrafficArea*, *function* describes, if the object for example may be a car driving lane, a pedestrian zones, or a cycle lane, while the *usage* attribute indicates which modes of transportation can use it (e.g. pedestrian, car, tram, roller skates). The attribute *surfaceMaterial* specifies the type of pavement and may also be used for *AuxiliaryTrafficAreas* (e.g. asphalt, concrete, gravel, soil, rail, grass). The *function* attribute of the *AuxiliaryTrafficArea* defines, for example kerbstones, middle lanes, or green areas. The possible values can also be specified in code lists.

The shape of each traffic area is defined by a surface geometry. Additional metadata may be defined by using attributes from pre-defined catalogues. This affects the class, function and usage of each traffic area as well as its surface material. The attribute catalogues may be customer- or country-specific. The following tables show examples for various kinds of *TrafficArea*:

Example:	Country road	Motorway entry
TransportationComplex – Function	road	road
TrafficArea – Usage	car, truck, bus, taxi, motorcycle	car, truck, bus, taxi, motorcycle
TrafficArea – Function	driving lane	motorway_entry
TrafficArea – SurfaceMaterial	asphalt	concrete

Example:	Runway of an airport	Apron of an airport
TransportationComplex – Function	road	apron
TrafficArea – Usage	aeroplane	aeroplane, car, truck, bus, pedestrian
TrafficArea – Function	airport – runway	airport – apron
TrafficArea – SurfaceMaterial	concrete	concrete

TransportationComplex is a subclass of *_TransportationObject* and of the root class *_CityObject*. The geometrical representation of the *TransportationComplex* varies through the different levels of detail. Since *TransportationComplex* is a subclass of *_CityObject* and hence a feature, it inherits the attribute *gml:name.* The street name is also stored within the *gml:name* property of the *Road* feature.

In the coarsest LOD0 the transportation complexes are modelled by line objects establishing a linear network. On this abstract level, path finding algorithms or similar analyses can be executed. It also can be used to generate schematic drawings and visualisations of the transport network. Since this abstract definition of transportation network does not contain explicit descriptions of the transportation objects, it may be task of the viewer application to generate the graphical visualisation, for example by using a library with style-definitions (width, color resp. texture) for each transportation object.

Starting from LOD1 a *TransportationComplex* provides an explicit surface geometry, reflecting the actual shape of the object, not just its centerline. In LOD2 to LOD4, it is further subdivided thematically into *TrafficAreas*, which are used by transportation, such as cars, trains, public transport, airplanes, bicycles or pedestrians and in *AuxiliaryTrafficAreas*, which are of minor importance for transportation purposes, for example road markings, green spaces or flower tubs. The different representations of a *TransportationComplex* for each LOD are illustrated in Fig. 60.

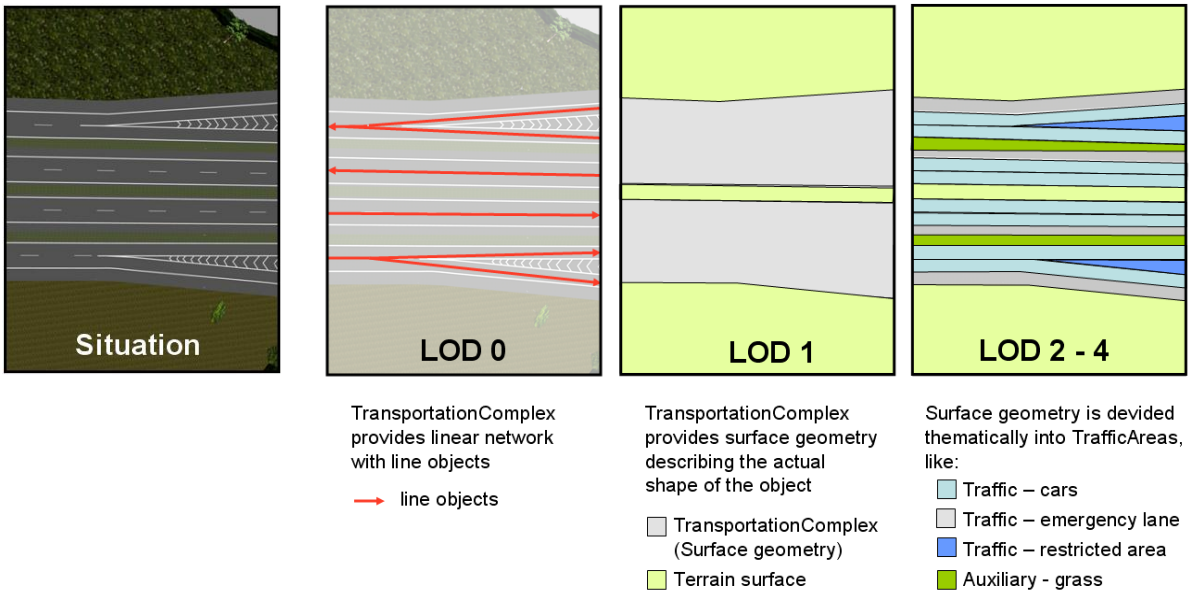

Fig. 60: *TransportationComplex* in LOD0, 1, and 2-4 (example shows part of a motorway) (source: Rheinmetall Defence Electronics).

In LOD0 areal transportation objects like squares should be modelled in the same way as in GDF, the ISO standard for transportation networks, which is used in most car navigation systems. In GDF a square is typically represented as a ring surrounding the place and to which the incident roads connect. CityGML does not cover further functional aspects of transportation network models (e.g. speed limits) as it is intended to complement and not replace existing standards like GDF. However, if specific functional aspects have to be associated with CityGML transportation objects, *generic attributes* provided by CityGML's *Generics* module (cf. chapter 10.12) can be used. Moreover, further objects of interest can be added from other information systems by the use of *ExternalReferences* (see chapter 6.11). For example, GDF datasets, which provide additional information for car navigation, can be used for simulation and visualisation of traffic flows. The values of the object attributes can be augmented or replaced using the concept of dictionaries (see chapter 6.6). These directories may be country- or user-specific (especially for country-specific road signs and signals).

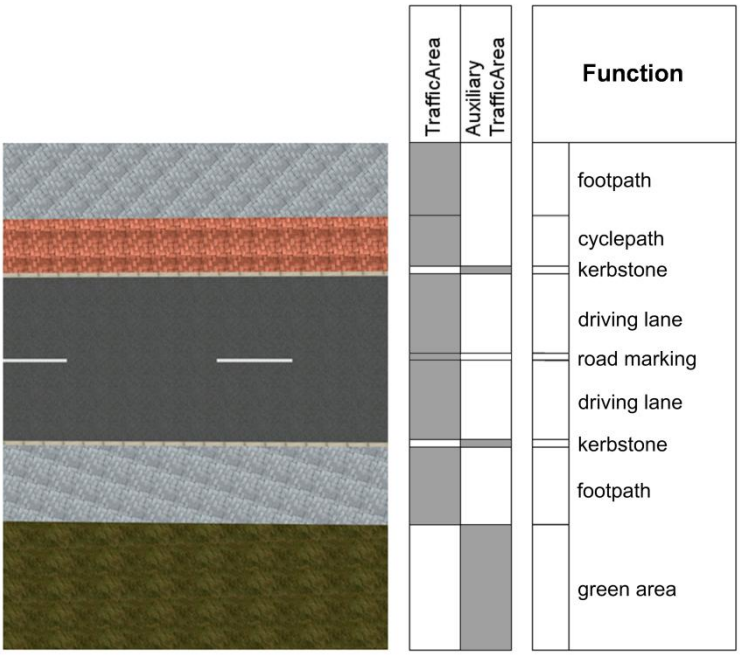

Fig. 61: *TransportationComplex* in LOD 2-4: representation of a road with a complex cross-section profile (example shows urban road) (source: Rheinmetall Defence Electronics).

The following example shows a complex urban crossing. The picture on the left is a screenshot of an editor application for a training simulator, which allows the definition of road networks consisting of transportation objects, external references, buildings and vegetation objects. On the right, the 3D representation of the defined crossing is shown including all referenced static and dynamic models.

Fig. 62: Complex urban intersection (left: linear transportation network with surface descriptions and external references, right: generated scene) (source: Rheinmetall Defence Electronics).

XML namespace

The XML namespace of the CityGML *Transportation* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/transportation/2.0. Within the XML Schema definition of the *Transportation* module, this URI is also used to identify the default namespace.

10.7.1 Transportation complex

AbstractTransportationObjectType, _TransportationObject

```
<xs:complexType name="AbstractTransportationObjectType" abstract="true">
  <xs:complexContent>
```



```
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfTransportationObject" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_TransportationObject" type="AbstractTransportationObjectType" abstract="true"
            substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTransportationObject" type="xs:anyType" abstract="true"/>
```

_TransportationObject represents the abstract superclass for transportation objects. Future extensions of the CityGML transportation model shall be modelled as subclasses of this class.

TransportationComplexType, TransportationComplex

```
<xs:complexType name="TransportationComplexType">
  <xs:complexContent>
    <xs:extension base="AbstractTransportationObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="trafficArea" type="TrafficAreaPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="auxiliaryTrafficArea" type="AuxiliaryTrafficAreaPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod0Network" type="gml:GeometricComplexPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfTransportationComplex" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="TransportationComplex" type="TransportationComplexType" substitutionGroup="_TransportationObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTransportationComplex" type="xs:anyType" abstract="true"/>
```

This type and element describe transportation complexes like roads or railways which may be aggregated from different thematic components (traffic areas, e.g. pedestrian path, and auxiliary traffic areas). As a subclass of *_CityObject*, *TransportationComplex* inherits all attributes and relations, in particular an id, names, external references, and generalisation relations. Furthermore, it represents the superclass for thematically distinct types of transportation complexes.

10.7.2 Subclasses of transportation complexes

TrackType, Track

```
<xs:complexType name="TrackType">
  <xs:complexContent>
    <xs:extension base="TransportationComplexType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfTrack" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Track" type="TrackType" substitutionGroup="TransportationComplex"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTrack" type="xs:anyType" abstract="true"/>
```

A *Track* is a small path mainly used by pedestrians. It is a subclass of *TransportationComplex* and thus inherits all its attributes and relations.

RoadType, Road

```
<xs:complexType name="RoadType">
  <xs:complexContent>
    <xs:extension base="TransportationComplexType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfRoad" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="Road" type="RoadType" substitutionGroup="TransportationComplex"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfRoad" type="xs:anyType" abstract="true"/>
```

Road is intended to be used to represent transportation features that are mainly used by vehicles like cars, for example streets, motorways, and country roads. It is a subclass of *TransportationComplex* and thus inherits all its attributes and relations.

RailwayType, Railway

```
<xs:complexType name="RailwayType">
  <xs:complexContent>
    <xs:extension base="TransportationComplexType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfRailway" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="Railway" type="RailwayType" substitutionGroup="TransportationComplex"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfRailway" type="xs:anyType" abstract="true"/>
```

Railway represents routes that are utilised by rail vehicles like trams or trains. It is a subclass of *TransportationComplex* and thus inherits all its attributes and relations.

SquareType, Square

```
<xs:complexType name="SquareType">
  <xs:complexContent>
    <xs:extension base="TransportationComplexType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfSquare" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="Square" type="SquareType" substitutionGroup="TransportationComplex"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfSquare" type="xs:anyType" abstract="true"/>
```

A *Square* is an open area commonly found in cities (e.g. a plaza, market square). It is a subclass of *TransportationComplex* and thus inherits all its attributes and relations.

10.7.3 Subdivisions of transportation complexes

TrafficAreaType, TrafficArea

```
<xs:complexType name="TrafficAreaType">
  <xs:complexContent>
    <xs:extension base="AbstractTransportationObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="surfaceMaterial" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfTrafficArea" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="TrafficArea" type="TrafficAreaType" substitutionGroup="_TransportationObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTrafficArea" type="xs:anyType" abstract="true"/>
```

AuxiliaryTrafficAreaType, AuxiliaryTrafficArea

```
<xs:complexType name="AuxiliaryTrafficAreaType">
  <xs:complexContent>
    <xs:extension base="AbstractTransportationObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="surfaceMaterial" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfAuxiliaryTrafficArea" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="AuxiliaryTrafficArea" type="AuxiliaryTrafficAreaType" substitutionGroup="_TransportationObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfAuxiliaryTrafficArea" type="xs:anyType" abstract="true"/>
```

10.7.4 Code lists

The attributes *class*, *function*, and *usage* of the features *TransportationComplex*, *TrafficArea* and *AuxiliaryTrafficArea* as well as the attribute *surfaceMaterial* of the features *TrafficArea* and *AuxiliaryTrafficArea* are specified as *gml:CodeType*. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.8.

10.7.5 Conformance requirements

Base requirements

1. For LOD0, the geometry of a *TransportationComplex* shall be modelled using GML line objects representing the centerline of the transportation complex. The line objects shall establish a linear network. Thus, the *lod0Network* property (type: *gml:GeometricComplexPropertyType*) of the element *TransportationComplex* may only contain or reference an appropriate curve geometry element.
2. Starting from LOD2, the *trafficArea* property (type: *TrafficAreaPropertyType*) as well as the *auxiliaryTrafficArea* property (type: *AuxiliaryTrafficAreaPropertyType*) of the element *TransportationCom-*

plex may be used. These properties shall not be used if the transportation complex is only represented in lower LODs.

Referential integrity

3. The *trafficArea* property (type: *TrafficAreaPropertyType*) of the element *TransportationComplex* may contain a *TrafficArea* element inline or an XLink reference to a remote *TrafficArea* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *trafficArea* property may only point to a remote *TrafficArea* element (where remote *TrafficArea* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
4. The *auxiliaryTrafficArea* property (type: *TrafficAreaPropertyType*) of the element *TransportationComplex* may contain an *AuxiliaryTrafficArea* element inline or an XLink reference to a remote *AuxiliaryTrafficArea* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *auxiliaryTrafficArea* property may only point to a remote *AuxiliaryTrafficArea* element (where remote *AuxiliaryTrafficArea* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

10.8 Vegetation objects

Vegetation features are important components of a 3D city model, since they support the recognition of the surrounding environment. By the analysis and visualisation of vegetation objects, statements on their distribution, structure and diversification can be made. Habitats can be analysed and impacts on the fauna can be derived. The vegetation model may be used as a basis for simulations of, for example forest fire, urban aeration or micro climate. The model could be used, for example to examine forest damage, to detect obstacles (e.g. concerning air traffic) or to perform analysis tasks in the field of environmental protection. The vegetation model of CityGML is defined by the thematic extension module *Vegetation* (cf. chapter 7).

The vegetation model of CityGML distinguishes between solitary vegetation objects like trees and vegetation areas, which represent biotopes like forests or other plant communities (Fig. 63). Single vegetation objects are modelled by the class *SolitaryVegetationObject*, whereas for areas filled with a specific vegetation the class *PlantCover* is used. The geometry representation of a *PlantCover* feature may be a *MultiSurface* or a *MultiSolid*, depending on the vertical extent of the vegetation. For example regarding forests, a *MultiSolid* representation might be more appropriate. The UML diagram of the vegetation model is depicted in Fig. 64, for the XML schema definition see below and annex A.12.

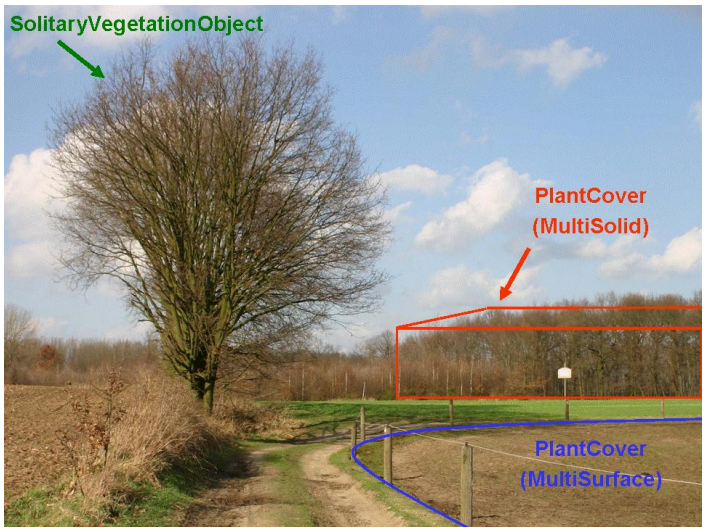

Fig. 63: Example for vegetation objects of the classes *SolitaryVegetationObject* and *PlantCover* (graphic: District of Recklinghausen).

A *SolitaryVegetationObject* may have the attributes *class*, *function*, *usage*, *species height*, *trunkDiameter* and *crownDiameter*. The attribute *class* contains the classification of the object or plant habit, e.g. tree, bush, grass, and can occur only once (see chapter 10.8.4 and annex C.7). The attribute *species* defines the species' name, for example "Abies alba", and can occur at most once (see chapter 10.8.4 and annex C.7). The optional attributes *function* and *usage* denotes the intended respectively real purpose of the object, for example botanical monument, and can occur multiple times. The possible attribute values for *class*, *species*, *function*, and *usage* can be provided in a code list. The attribute *height* contains the relative height of the object. The attributes *crownDiameter* and *trunkDiameter* represent the plant crown and trunk diameter respectively. The trunk diameter is often used in regulations of municipal cadastre (e.g. tree management rules).

A *PlantCover* feature may have the attributes *class*, *function*, *usage* and *averageHeight*. The plant community of a *PlantCover* is represented by the attribute *class*. The values of this attribute can be specified in a code list (cf. chapter 10.8.4 and annex C.7) whose values should not only describe one plant type or species, but denote a typical mixture of plant types in a plant community. This information can be used in particular to generate realistic 3D visualisations, where the *PlantCover* region is automatically, perhaps randomly, filled with a corresponding mixture of 3D plant objects. The attributes *function* and *usage* indicate the intended respectively real purpose of the object, for example national forest, and can occur multiple times. The attribute *averageHeight* denotes the average relative vegetation height.

Since both *SolitaryVegetationObject* and *PlantCover* are derived fom *_CityObject*, they inherit all attributes of a city object, in particular a name (*gml:name)* and an *ExternalReference* to a corresponding object in an external information system, which may contain botanical information from public environmental agencies (see chapter 6.7).

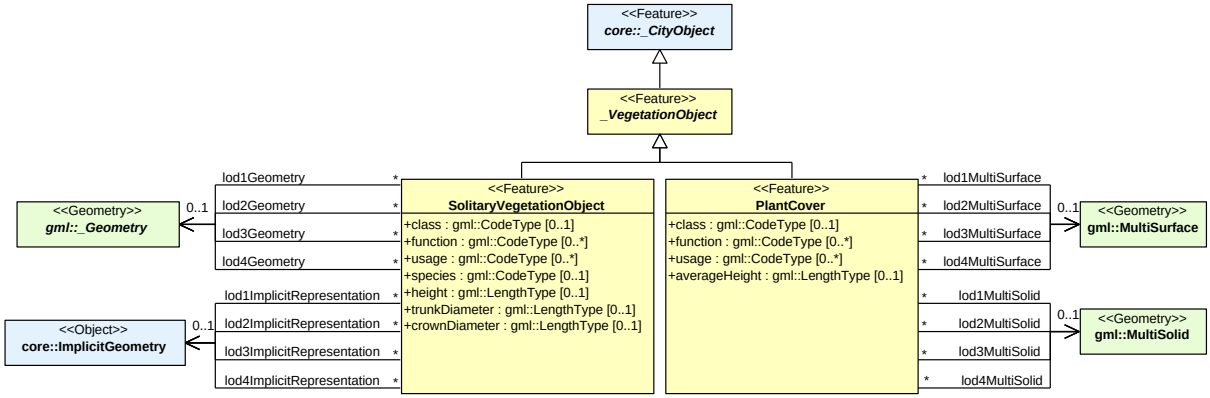

Fig. 64: UML diagram of vegetation objects in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Vegetation* module.

The geometry of a *SolitaryVegetationObject* may be defined in LOD 1-4 explicitly by a GML geometry having absolute coordinates, or prototypically by an *ImplicitGeometry* (cf. chapter 8.2). Solitary vegetation objects probably are one of the most important features where implicit geometries are appropriate, since the shape of most types of vegetation objects, such as trees of the same species, can be treated as identical in most cases. Furthermore, season dependent appearances may by mapped using *ImplicitGeometry*. For visualisation purposes, only the content of the library object defining the object's shape and appearance has to be swapped (cf. Fig. 65).

Fig. 65: Visualisation of a vegetation object in different seasons (source: District of Recklinghausen).

A *SolitaryVegetationObject* or a *PlantCover* may have a different geometry in each LOD. Whereas a *SolitaryVegetationObject* is associated with the *gml:_Geometry* class representing an arbitrary GML geometry (by the relation *lodXGeometry*, X ∈ [1..4]), a *PlantCover* is restricted to be either a *gml:MultiSolid* or a *gml:MultiSurface.* An example of a *PlantCover* modelled as *gml:MultiSolid* is a '*solid forest model'*, see Fig. 66.

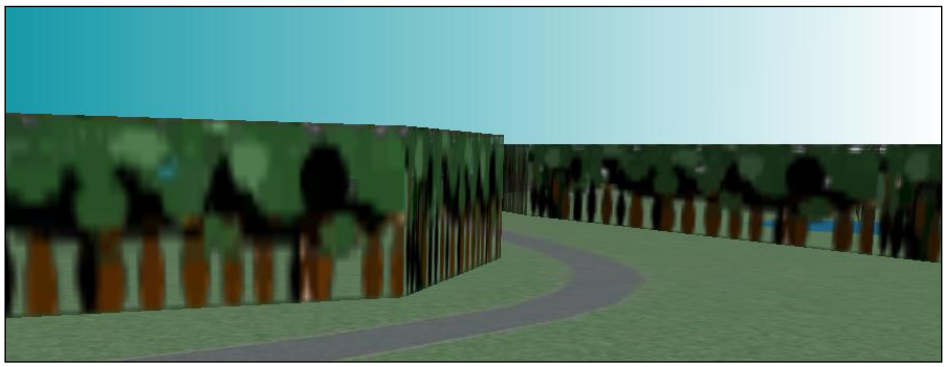

Fig. 66: Example for the visualisation/modelling of a solid forest (source: District of Recklinghausen).

XML namespace

The XML namespace of the CityGML *Vegetation* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/vegetation/2.0. Within the XML Schema definition of the *Vegetation* module, this URI is also used to identify the default namespace.

10.8.1 Vegetation object

AbstractVegetationObjectType, _VegetationObject

```
<xs:complexType name="AbstractVegetationObjectType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfVegetationObject" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_VegetationObject" type="AbstractVegetationObjectType" abstract="true" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfVegetationObject" type="xs:anyType" abstract="true"/>
```

10.8.2 Solitary vegetation objects

SolitaryVegetationObjectType, SolitaryVegetationObject

```
<xs:complexType name="SolitaryVegetationObjectType">
  <xs:complexContent>
    <xs:extension base="AbstractVegetationObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="species" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="height" type="gml:LengthType" minOccurs="0"/>
        <xs:element name="trunkDiameter" type="gml:LengthType" minOccurs="0"/>
        <xs:element name="crownDiameter" type="gml:LengthType" minOccurs="0"/>
        <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfSolitaryVegetationObject" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
```

```
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="SolitaryVegetationObject" type="SolitaryVegetationObjectType" substitutionGroup="_VegetationObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfSolitaryVegetationObject" type="xs:anyType" abstract="true"/>
```

10.8.3 Plant cover objects

PlantCoverType, PlantCover

```
<xs:complexType name="PlantCoverType">
  <xs:complexContent>
    <xs:extension base="AbstractVegetationObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="averageHeight" type="gml:LengthType" minOccurs="0"/>
        <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod1MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0"/>
        <xs:element name="lod2MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0"/>
        <xs:element name="lod3MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfPlantCover" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="PlantCover" type="PlantCoverType" substitutionGroup="_VegetationObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfPlantCover" type="xs:anyType" abstract="true"/>
```

10.8.4 Code lists

The attributes *class*, *function*, and *usage* of the features *PlantCover* and *SolitaryVegetationObject* as well as the attribute *species* of the feature *SolitaryVegetationObject* are specified as *gml:CodeType*. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.7.

10.8.5 Example CityGML dataset

The following two excerpts of a CityGML dataset contain a solitary tree (*SolitaryVegetationObject)* and a plant community (*PlantCover*). The solitary tree has the attributes: *class* = 1070 (deciduous tree), *species* = 1040 (Fagus/beech), *height* = 8 m, *trunkDiameter* = 0.7 m, *crownDiameter* = 8.0 m. The plant community has the attributes: *class* =1180 (isoeto-nanojuncetea), *averageHeight* = 0.5 m. The attribute values of the *class* and *species* attributes are taken from code lists proposed by the SIG 3D which are presented in annex C.7.

```
<SolitaryVegetationObject>
  <class codeSpace="http://www.sig3d.org/codelists/standard/vegetation/2.0/SolitaryVegetationObject_class.xml">1070</class>
  <species codeSpace="http://www.sig3d.org/codelists/standard/vegetation/2.0/SolitaryVegetationObject_species.xml">1040</species>
  <height uom="#m">8</height>
  <trunkDiameter uom="#m">0.7</trunkDiameter>
  <crownDiameter uom="#m">8</crownDiameter>
  <lod1ImplicitRepresentation>
    <core:ImplicitGeometry>
      <core:mimeType>1010</core:mimeType>
      <core:libraryObject>urn:sig3d:tree.wrl</core:libraryObject>
      <core:referencePoint>
        <gml:Point srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
          <gml:pos srsDimension="3">5733690.578 2571129.123 60.0</gml:pos>
        </gml:Point>
      </core:referencePoint>
    </core:ImplicitGeometry>
```



```
  </lod1ImplicitRepresentation>
</SolitaryVegetationObject>
```

```
<PlantCover>
  <class codeSpace="http://www.sig3d.org/codelists/standard/vegetation/2.0/PlantCover_class.xml">1180</class>
  <averageHeight uom="#m">0.5</averageHeight>
  <lod1MultiSurface>
    <gml:MultiSurface>
      <gml:surfaceMember>
        <gml:Polygon srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
          <gml:exterior>
            <gml:LinearRing>
              <gml:pos srsDimension="3">5733806.146 2571329.227 60.0</gml:pos>
              <gml:pos srsDimension="3">5733754.782 2571387.011 60.0</gml:pos>
              <gml:pos srsDimension="3">5733674.527 2571374.170 60.0</gml:pos>
              <gml:pos srsDimension="3">5733670.246 2571274.653 60.0</gml:pos>
              <gml:pos srsDimension="3">5733764.413 2571243.621 60.0</gml:pos>
              <gml:pos srsDimension="3">5733806.146 2571329.227 60.0</gml:pos>
            </gml:LinearRing>
          </gml:exterior>
        </gml:Polygon>
      </gml:surfaceMember>
    </gml:MultiSurface>
    …
  </lod1MultiSurface>
</PlantCover>
```

10.8.6 Conformance requirements

Referential integrity

- The *lodXImplicitRepresentation*, X ∈ [1..4], property (type: *core:ImplicitRepresentationPropertyType*) of the element *SolitaryVegetationObject* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lodXImplicitRepresentation*, X ∈ [1..4], property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

10.9 City furniture

City furniture objects are immovable objects like lanterns, traffic lights, traffic signs, flower buckets, advertising columns, benches, delimitation stakes, or bus stops (Fig. 67, Fig. 68). City furniture objects can be found in traffic areas, residential areas, on squares or in built-up areas. The modelling of city furniture objects is used for visualisation of, for example city traffic, but also for analysing local structural conditions. The recognition of special locations in a city model is improved by the use of these detailed city furniture objects, and the city model itself becomes more alive and animated. The city furniture model of CityGML is defined by the thematic extension module *CityFurniture* (cf. chapter 7).

City furniture objects can have an important influence on simulations of, for example city traffic situations. Navigation systems can be realised, for example for visually handicapped people using a traffic light as routing target. Likewise, city furniture objects are important to plan a heavy vehicle transportation, where the exact position and further conditions of obstacles must be known.

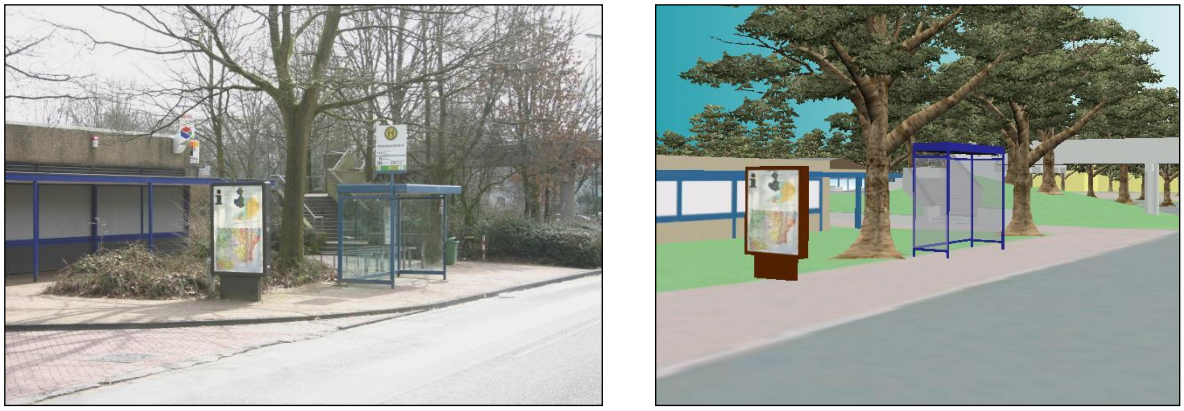

Fig. 67: Real situation showing a bus stop (left). The advertising billboard and the refuge are modelled as *CityFurniture* objects in the right image (source: 3D city model of Barkenberg).

Fig. 68: Real situation showing lanterns and delimitation stakes (left). In the right image they are modelled as *CityFurniture* objects with *ImplicitGeometry* representations (source: 3D city model of Barkenberg).

The UML diagram of the city furniture model is depicted in Fig. 69, for the XML schema definition see below and annex A.5.

The class *CityFurniture* may have the attributes *class, function* and *usage*. Their possible values can be specified in corresponding code lists (chapter 10.9.2 and annex C.4). The *class* attribute allows an object classification like traffic light, traffic sign, delimitation stake, or garbage can, and can occur only once. The *function* attribute describes, to which thematic area the city furniture object belongs to (e.g. transportation, traffic regulation, architecture), and can occur multiple times. The attribute *usage* denotes the real purpose of the object.

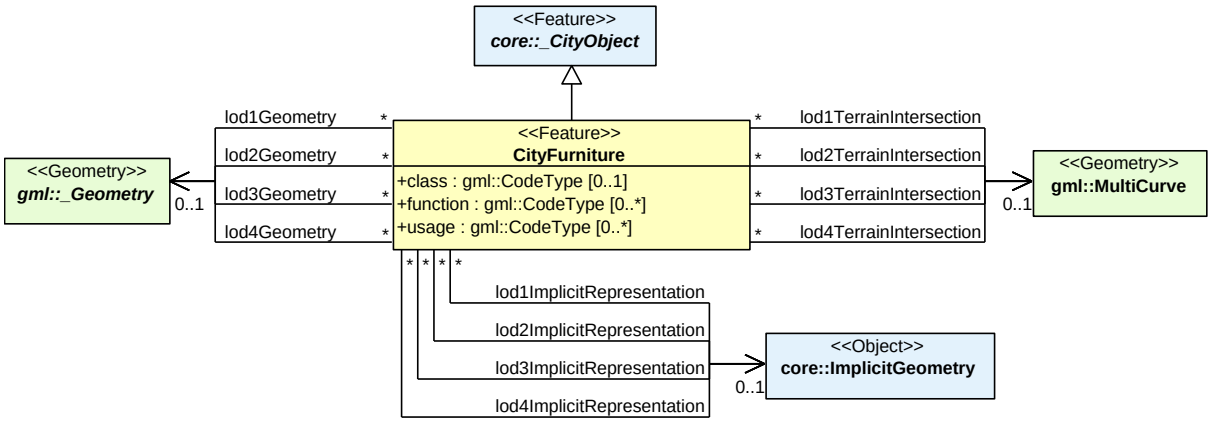

Fig. 69: UML diagram of city furniture objects in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *CityFurniture* module.

Since *CityFurniture* is a subclass of *_CityObject* and hence is a feature, it inherits the attribute *gml:name*. As with any *_CityObject*, *CityFurniture* objects may be assigned *ExternalReferences* (cf. chapter 6.7) and may be augmented by *generic attributes* using CityGML's *Generics* module (cf. chapter 10.12). For *ExternalReferences* city furniture objects can have links to external thematic databases. Thereby, semantic information of the objects, which cannot be modelled in CityGML, can be transmitted and used in the 3D city model for further processing, for example information from systems of powerlines or pipelines, traffic sign cadaster, or water resources for disaster management.

City furniture objects can be represented in city models with their specific geometry, but in most cases the same kind of object has an identical geometry. The geometry of *CityFurniture* objects in LOD 1-4 may be represented by an explicit geometry (*lodXGeometry* where *X* is between 1 and 4) or an *ImplicitGeometry* object (*lodXImplicitRepresentation* with *X* between 1 and 4). Following the concept of *ImplicitGeometry* the geometry of a prototype city furniture is stored only once in a local coordinate system and referenced by other city furniture features (see chapter 8.2). Spatial information of city furniture objects can be taken, for example, from city maps or from public and private external information systems.

In order to specify the exact intersection of the DTM with the 3D geometry of a city furniture object, the latter can have a *TerrainIntersectionCurve* (TIC) for each LOD (cf. chapter 6.5). This allows for ensuring a smooth transition between the DTM and the city furniture object.

XML namespace

The XML namespace of the CityGML *CityFurniture* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/cityfurniture/2.0. Within the XML Schema definition of the *CityFurniture* module, this URI is also used to identify the default namespace.

10.9.1 City furniture object

CityFurnitureType, CityFurniture

```
<xs:complexType name="CityFurnitureType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
```



```
        <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfCityFurniture" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="CityFurniture" type="CityFurnitureType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfCityFurniture" type="xs:anyType" abstract="true"/>
```

10.9.2 Code lists

The attributes *class*, *function*, and *usage* of the feature *CityFurniture* are specified as *gml:CodeType*. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.4.

10.9.3 Example CityGML dataset

The following example of a CityGML dataset is an extract of the model of a delimitation stake in LOD3 and contains the attributes *class* = 1000 and *function* = 1520 (delimitation stake) whose coded attribute values are taken from a code list proposed by the SIG 3D (cf. annex C.4). The delimitation stake with the object ID *stake0815* has an *ExternalReference* pointing to a cadastre object within the German ALKIS database (www.adv-online.de) which is identified by the URI *urn:adv:oid:DEHE123400007001*.

The example dataset shows the geometry of the top surface (*gml:id* "cover") and of the left surface (*gml:id* "surfLeft") of the stake which are both depicted in Fig. 70. The top surface is assigned a constant material (white color) and the left surface is textured using the texture image *stake.gif* by denoting the relevant texture coordinates. Both surface appearances are modelled using the CityGML *Appearance* module (cf. chapter 9).

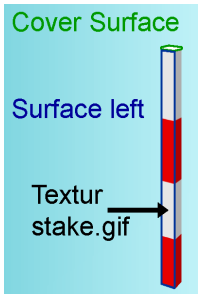

Fig. 70: Example of a simple city furniture object (source: District of Recklinghausen).

```
<!-- delimitation stake in LOD3 -->
<CityFurniture gml:id="stake0815">
  <core:externalReference>
    <core:informationSystem>http://www.adv-online.de</core:informationSystem>
    <!-- Reference to ALKIS -->
    <core:externalObject>
      <core:uri>urn:adv:oid:DEHE123400007001</core:uri>
      <!-- ALKIS Object ID -->
    </core:externalObject>
  </core:externalReference>
  <app:appearance>
    <app:Appearance>
      <app:surfaceDataMember>
        <app:X3DMaterial>
          <app:ambientIntensity>0.4</app:ambientIntensity>
          <app:diffuseColor>1.0  1.0  1.0</app:diffuseColor>
          <app:target>#cover</app:target>
        </app:X3DMaterial>
      </app:surfaceDataMember>
      <app:surfaceDataMember>
        <app:ParameterizedTexture>
          <app:imageURI>stake.gif</app:imageURI>
```



```
            <app:textureType>typical</app:textureType>
            <app:target uri="#surfLeft">
              <app:TexCoordList>
                <app:textureCoordinates ring="#surfLeft_ring1">
                  0.000 0.000 1.000 0.000 1.000 1.000 0.000 1.000 0.000 0.000
                </app:textureCoordinates>
              </app:TexCoordList>
            </app:target>
          </app:ParameterizedTexture>
        </app:surfaceDataMember>
      </app:Appearance>
    </app:appearance>
    <class codeSpace="http://www.sig3d.org/codelists/standard/cityfurniture/2.0/CityFurniture_class.xml">1000</class>
    <!-- 1000 = traffic  -->
    <function codeSpace="http://www.sig3d.org/codelists/standard/cityfurniture/2.0/CityFurniture_function.xml">1520</function>
    <!-- 1520 = boundary post  -->
    <lod3Geometry>
      <!--Stake 0.06x0.06x1.2-->
      <gml:Solid srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
        <gml:exterior>
          <gml:CompositeSurface>
            <gml:surfaceMember>
            <gml:Polygon gml:id="cover">
                <gml:exterior>
                  <!-- Cover-Surface  -->
                  <gml:LinearRing>
                    <gml:pos>5733500.060 2572400.060 61.200</gml:pos>
                    <gml:pos>5733500.060 2572400.000 61.200</gml:pos>
                    <gml:pos>5733500.000 2572400.000 61.200</gml:pos>
                    <gml:pos>5733500.000 2572400.060 61.200</gml:pos>
                    <gml:pos>5733500.060 2572400.060 61.200</gml:pos>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
            <gml:surfaceMember>
              <gml:Polygon gml:id="surfLeft">
                <gml:exterior>
                  <!-- Surface left  -->
                  <gml:LinearRing gml:id="surfLeft_ring1">
                    <gml:pos>5733500.060 2572400.000 60.000</gml:pos>
                    <gml:pos>5733500.000 2572400.000 60.000</gml:pos>
                    <gml:pos>5733500.000 2572400.000 61.200</gml:pos>
                    <gml:pos>5733500.060 2572400.000 61.200</gml:pos>
                    <gml:pos>5733500.060 2572400.000 60.000</gml:pos>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
            ...
          </gml:CompositeSurface>
        </gml:exterior>
      </gml:Solid>
    </lod3Geometry>
</CityFurniture>
```

10.9.4 Conformance requirements

Referential integrity

- The *lodXImplicitRepresentation*, X ∈ [1..4], property (type: *core:ImplicitRepresentationPropertyType*) of the element *CityFurniture* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lodXImplicitRepresentation*, X ∈ [1..4], property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

10.10 Land use

LandUse objects can be used to describe areas of the earth's surface dedicated to a specific land use, but also to describe areas of the earth's surface having a specific land cover with or without vegetation, such as sand, rock, mud flats, forest, grasslands, etc (i.e. the physical appearance). Land use and land cover are different concepts; the first describes human activities on the earth's surface, the second describes its physical and biological cover. However, the two are interlinked and often mixed in practice. *LandUse* objects in CityGML represent both concepts: They can be employed to represent, parcels, spatial planning objects, recreational objects and objects describing the physical characteristics of an area, in 3D (e.g. wetlands). Fig. 71 shows the UML diagram of land use objects, for the XML schema definition see chapter 10.10.1 and annex A.8. The land use model of CityGML is provided by the thematic extension module *LandUse* (cf. chapter 7).

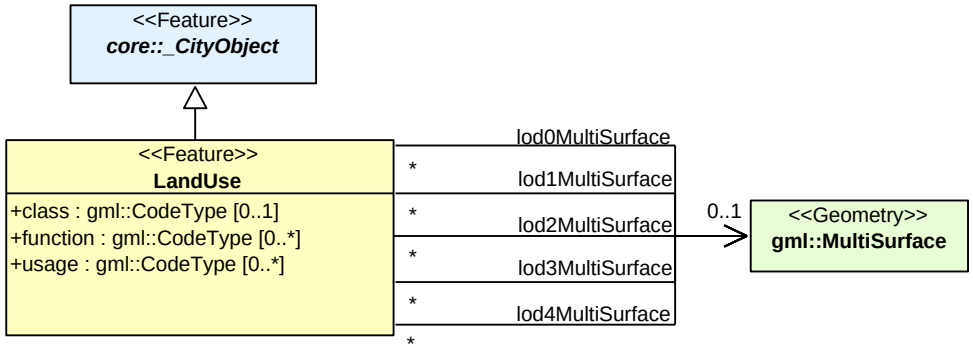

Fig. 71: UML diagram of land use objects in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *LandUse* module.

Every *LandUse* object may have the attributes *class*, *function*, and *usage*. The *class* attribute is used to represent the classification of land use objects, like settlement area, industrial area, farmland etc., and can occur only once. The possible values can be specified in a code list (cf. annex C.5). The attribute *function* defines the purpose of the object or their nature, like e.g. cornfield or heath, while the attribute *usage* can be used, if the way the object is actually used differs from the function. Both attributes can occur multiple times.

The *LandUse* object is defined for all LOD 0-4 and may have different geometries in any LOD. The surface geometry of a *LandUse* object is required to have 3D coordinate values. It must be a GML3 *MultiSurface*, which might be assigned appearance properties like textures or colors (using CityGML's appearance model, cf. chapter 9).

LandUse objects can be employed to establish a coherent geometric/semantical tesselation of the earth's surface. In this case topological relations between neighbouring *LandUse* objects should be made explicit by defining the boundary *LineStrings* only once and by referencing them in the corresponding *Polygons* using XLinks (cf. chapter 8.1). Fig. 72 shows a land use tesselation, where the geometries of the land use objects are represented as triangulated surfaces. In fact, they are the result of a constrained triangulation of a DTM with consideration of breaklines defined by a 2D vector map of land use classifications.

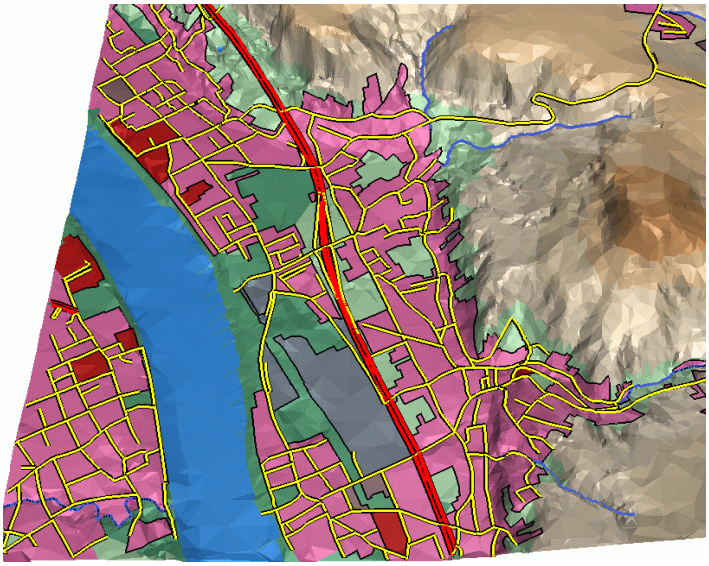

Fig. 72: LOD0 regional model consisting of land use objects in CityGML (source: IGG Uni Bonn).

XML namespace

The XML namespace of the CityGML *LandUse* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/landuse/2.0. Within the XML Schema definition of the *LandUse* module, this URI is also used to identify the default namespace.

10.10.1 Land use object

LandUseType, LandUse

```
<xs:complexType name="LandUseType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod0MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfLandUse" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="LandUse" type="LandUseType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfLandUse" type="xs:anyType" abstract="true"/>
```

10.10.2 Code lists

The attributes *class*, *function*, and *usage* of the feature *LandUse* are specified as *gml:CodeType*. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.5.

10.10.3 Conformance requirements

Base requirements

- The *gml:MultiSurface* geometry element representing the geometry of a *LandUse* object must be given with 3D coordinate values within each LOD.

10.11 City object groups

The CityGML grouping concept has been introduced in chapter 6.8. *CityObjectGroups* are modelled using the Composite Design Pattern from software engineering (cf. Gamma et al. 1995): *CityObjectGroups* aggregate *CityObjects* and furthermore are defined as special *CityObjects*. This implies that a group may become a member of another group realizing a recursive aggregation schema. However, in a CityGML instance document it has to be ensured (by the generating application) that no cyclic groupings are included. Fig. 73 shows the UML diagram for the class *CityObjectGroup*, for the XML schema see annex A.6. The grouping concept of CityGML is defined by the thematic extension module *CityObjectGroup* (cf. chapter 7).

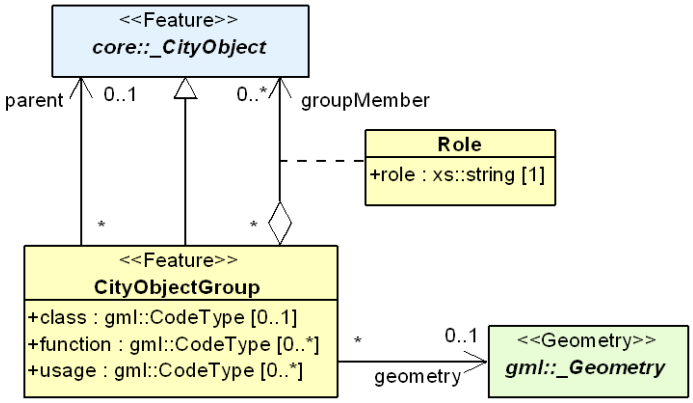

Fig. 73: UML diagram of city object groups in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *CityObjectGroup* module.

The class *CityObjectGroup* has the optional attributes *class*, *function* and *usage*. The *class* attribute allows a group classification with respect to the stated function and may occur only once. The *function* attribute is intended to express the main purpose of a group, possibly to which thematic area it belongs (e.g. site, building, transportation, architecture, unknown etc.). The attribute *usage* can be used, if the way the object is actually used differs from the function. Both attributes can occur multiple times. Each member of a group may be qualified by a *role* name, reflecting the role each *_CityObject* plays in the context of the group. Furthermore, a *CityObjectGroup* can optionally be assigned an arbitrary geometry object from the GML3 subset shown in Fig. 9 in chapter 8.1. This may be used to represent a generalised geometry generated from the members geometries.

The parent association linking a *CityObjectGroup* to a *_CityObject* allows for the modelling of a generic hierarchical grouping concept. Named aggregations of components (CityObjects) can be added to specific CityObjects considered as the parent object. The parent association links to the aggregate, while the parts are given by the group members. This concept is used, for example, to represent storeys in buildings (see section 10.3.6: Modelling building storeys using CityObjectGroups).

XML namespace

The XML namespace of the CityGML *CityObjectGroup* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/cityobjectgroup/2.0. Within the XML Schema definition of the *CityObjectGroup* module, this URI is also used to identify the default namespace.

10.11.1 City object group

CityObjectGroupType, CityObjectGroup

```
<xs:complexType name="CityObjectGroupType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="groupMember" type="CityObjectGroupMemberType" minOccurs="0" maxOccurs="unbounded"/>
```

```
        <xs:element name="parent" type="CityObjectGroupParentType" minOccurs="0"/>
        <xs:element name="geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfCityObjectGroup" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="CityObjectGroup" type="CityObjectGroupType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfCityObjectGroup" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="CityObjectGroupMemberType">
  <xs:sequence minOccurs="0">
    <xs:element ref="core:_CityObject"/>
  </xs:sequence>
  <xs:attribute name="role" type="xs:string"/>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:complexType name="CityObjectGroupParentType">
  <xs:sequence minOccurs="0">
    <xs:element ref="core:_CityObject"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
```

10.11.2 Code lists

The attributes *class*, *function*, and *usage* of the feature *CityObjectGroup* are specified as *gml:CodeType*. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.10.

10.11.3 Conformance requirements

Base requirements

1. No cyclic groupings shall be included within a CityGML instance document.

Referential integrity

2. The *groupMember* property (type: *CityObjectGroupMemberType*) of the element *CityObjectGroup* may contain a *core:_CityObject* element inline or an XLink reference to a remote *core:_CityObject* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *groupMember* property may only point to a remote *core:_CityObject* element (where remote *core:_CityObject* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
3. The *parent* property (type: *CityObjectGroupParentType*) of the element *CityObjectGroup* may contain a *core:_CityObject* element inline or an XLink reference to a remote *core:_CityObject* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *parent* property may only point to a remote *core:_CityObject* element (where remote *core:_CityObject* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

10.12 Generic city objects and attributes

The concept of generic city objects and attributes allows for the storage and exchange of 3D objects which are not covered by any explicitly modelled thematic class within CityGML or which require attributes not represented in CityGML. These generic extensions to the CityGML data model are realised by the classes *GenericCityObject* and *_genericAttribute* defined within the thematic extension module *Generics* (cf. chapter 7). In order to avoid problems concerning semantic interoperability, generic extensions shall only be used if appropriate thematic classes or attributes are not provided by any other CityGML module.

Fig. 74 shows the UML diagram of generic objects and attributes. For XML schema definition see below and annex A.7.

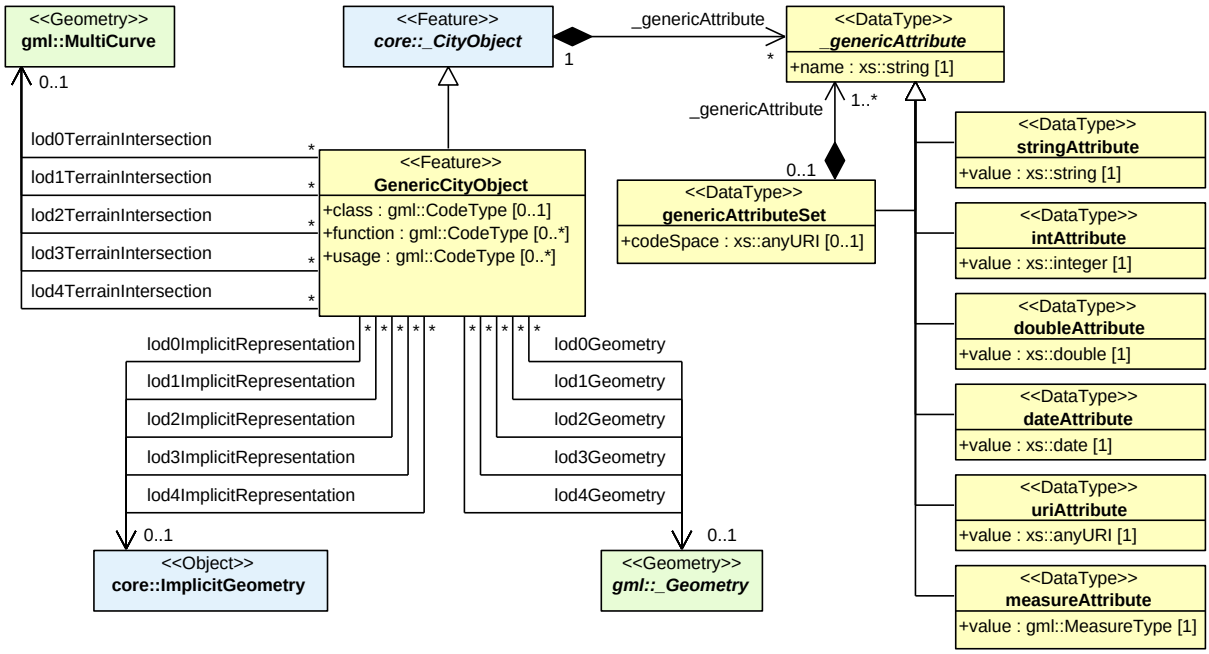

Fig. 74: UML diagram of generic objects and attributes in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Generics* module.

Generic attributes are name-value pairs associated with a city object. Each generic attribute has a mandatory *name* identifier which can be freely chosen. The data type of the attribute value may be *String*, *Integer*, *Double* (floating point number), *URI*, *Date*, and *gml:MeasureType*. The attribute type is defined by the selection of the particular subclass of *_genericAttribute*, for example *stringAttribute*, *intAttribute*, etc. A *measureAttribute* facilitates the representation of measured values. Its value is of the structured type *gml:MeasureType* which provides an optional attribute *uom* (units of measure) of type *xs:anyURI* that points to a reference system for the amount.

Generic attributes can be grouped under a common name using a *genericAttributeSet*. The *genericAttributeSet* class is derived from *_genericAttribute* and thus is also realized as generic attribute. Its value is the set of contained generic attributes and its *name* property provides a name identifier for the entire set. Since *genericAttributeSet* is itself a generic attribute, it may also be contained in a generic attribute set facilitating a recursive nesting of arbitrary depth. The optional *codeSpace* attribute (of type *xs:anyURI*) of *genericAttributeSet* is used to associate the attribute set with an authority, e.g. the organisation or community who defined the attribute set and its contained attributes. By this means, generic attribute sets can be clearly distinguished even if they share the same name.

In order to model generic attributes, the abstract base class *_CityObject* defined within the *CityGML Core* module is augmented by the additional property element *_genericAttribute* using CityGML's *Application Domain Extension* mechanism (cf. chapter 6.12). By this means, each thematic subclass of *_CityObject* inherits this property and, thus, may be assigned an arbitrary number of generic attributes in order to represent additional properties of features not represented by the explicitly modelled thematic classes of the CityGML data model.

Thus, the *Generics* module has a deliberate impact on all CityGML extension modules defining thematic sub-classes of *_CityObject*.

A *GenericCityObject* may have the attributes *class*, *function* and *usage* defined as *gml:CodeType*. The *class* attribute allows an object classification within the thematic area such as pipe, power line, dam, or unknown. The *function* attribute describes to which thematic area the *GenericCityObject* belongs (e.g. site, transportation, architecture, energy supply, water supply, unknown etc.). The attribute *usage* can be used, if the way the object is actually used differs from the function. Both attributes can occur multiple times.

The geometry of a *GenericCityObject* can either be an explicit GML3 geometry or an *ImplicitGeometry* (see chapter 8.2). In the case of an explicit geometry the object can have only one geometry for each LOD, which may be an arbitrary 3D GML geometry object (class *gml:_Geometry*, which is the base class of all GML geometries, *lodXGeometry*, X ∈ [0..4]). Absolute coordinates according to the reference system of the city model must be given for the explicit geometry. In the case of an *ImplicitGeometry*, a reference point (anchor point) of the object and optionally a transformation matrix must be given. In order to compute the actual location of the object, the transformation of the local coordinates into the reference system of the city model must be processed and the anchor point coordinates must be added. The shape of an *ImplicitGeometry* can be given as an external resource with a proprietary format, e.g. a VRML or DXF file from a local file system or an external web service. Alternatively the shape can be specified as a 3D GML3 geometry with local cartesian coordinates using the property *relativeGeometry* (further details are given in chapter 8.2).

In order to specify the exact intersection of the DTM with the 3D geometry of a *GenericCityObject*, the latter can have *TerrainIntersectionCurves* for every LOD (cf. chapter 6.5). This is important for 3D visualisation but also for certain applications like driving simulators. For example, if a city wall (e.g., the Great Wall of China) should be represented as a *GenericCityObject*, a smooth transition between the DTM and the city wall would have to be ensured.

XML namespace

The XML namespace of the CityGML *Generics* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/generics/2.0. Within the XML Schema definition of the *Generics* module, this URI is also used to identify the default namespace.

10.12.1 Generic city object

GenericCityObjectType, GenericCityObject

```
<xs:complexType name="GenericCityObjectType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod0Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod0TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod0ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="GenericCityObject" type="GenericCityObjectType" substitutionGroup="core:_CityObject"/>
```


10.12.2 Generic attributes

AbstractGenericAttributeType, _genericAttribute, StringAttributeType, stringAttribute, etc.

```
<xs:complexType name="AbstractGenericAttributeType" abstract="true">
  <xs:sequence/>
  <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_genericAttribute" type="AbstractGenericAttributeType" abstract="true"
            substitutionGroup="core:_GenericApplicationPropertyOfCityObject"/>
<!-- ========================================================================================== -->
<xs:complexType name="StringAttributeType">
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="stringAttribute" type="StringAttributeType" substitutionGroup="_genericAttribute"/>
<!-- ========================================================================================== -->
<xs:complexType name="IntAttributeType">
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:integer"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="intAttribute" type="IntAttributeType" substitutionGroup="_genericAttribute"/>
<!-- ========================================================================================== -->
<xs:complexType name="DoubleAttributeType">
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:double"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="doubleAttribute" type="DoubleAttributeType" substitutionGroup="_genericAttribute"/>
<!-- ========================================================================================== -->
<xs:complexType name="DateAttributeType">
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:date"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="dateAttribute" type="DateAttributeType" substitutionGroup="_genericAttribute"/>
<!-- ========================================================================================== -->
<xs:complexType name="UriAttributeType">
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="uriAttribute" type="UriAttributeType" substitutionGroup="_genericAttribute"/>
<!-- ========================================================================================== -->
<xs:complexType name="MeasureAttributeType">
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
```

```
      <xs:sequence>
        <xs:element name="value" type="gml:MeasureType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="measureAttribute" type="MeasureAttributeType" substitutionGroup="_genericAttribute"/>
```

GenericAttributeSetType, genericAttributeSet

```
<xs:complexType name="GenericAttributeSetType">
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <xs:sequence>
        <xs:element ref="_genericAttribute" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="codeSpace" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="genericAttributeSet" type="GenericAttributeSetType" substitutionGroup="_genericAttribute"/>
```

10.12.3 Code lists

The attributes *class*, *function*, and *usage* of the feature *GenericCityObject* are specified as *gml:CodeType*. The values of these properties can be enumerated in code lists.

10.12.4 Conformance requirements

Usage restriction for generic city objects and attributes

1. The *GenericCityObject* element may only be used to model thematic city objects which are not provided by any other CityGML module and, thus, are not covered by the overall CityGML data model. If an appropriate thematic class is available though, this thematic class shall be used instead, and, consequently, the corresponding CityGML module has to be employed by the CityGML instance document.
2. The *_genericAttribute* property of the element *core:_CityObject* shall only be used to describe additional properties of features not represented by the explicitly modelled thematic classes of the CityGML data model. Thus, generic attributes shall only be modelled if the appropriate thematic class representing the feature does not offer a feasible property.

Referential integrity

3. The *lodXImplicitRepresentation*, X ∈ [0..4], property (type: *core:ImplicitRepresentationPropertyType*) of the element *GenericCityObject* may contain a *core:ImplicitGeometry* element inline or an XLink reference to a remote *core:ImplicitGeometry* element using the XLink concept of GML 3.1.1. In the latter case, the *xlink:href* attribute of the *lodXImplicitRepresentation*, X ∈ [0..4], property may only point to a remote *core:ImplicitGeometry* element (where remote *core:ImplicitGeometry* elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

10.13 Application Domain Extensions (ADE)

CityGML has been designed as an application independent information model and exchange format for 3D city and landscape models. However, specific applications typically have additional information needs to be modelled and exchanged. In general, there are two different approaches to combine city model data and application data:

1. Embed the CityGML objects into a (larger) application framework and establish the connection between application data and CityGML data within the application framework. For example, CityGML data fragments may be embedded into the application's XML data files or stored as attributes of application objects according to the application's data model.
2. Incorporate application specific information into the CityGML instance documents. This approach is especially feasible, if the application specific information follows essentially the same structure as defined by the CityGML schema. This is the case, if the application data could be represented by additional attributes of CityGML objects and only some new feature types would have to be defined.

In the following, we will focus on the second option, as only this approach lies within the scope of this specification. Generic attributes and objects have already been introduced as a first possibility to support the exchange of application specific data (see section 10.12). Whereas they allow to extend CityGML without changing its XML schema definition, this flexibility has some disadvantages:

- Generic attributes and objects may occur arbitrarily in the CityGML instance documents, but there is no formal specification of the names, datatypes, and multiplicities. Thus, there is no guarantee for an application that a specific instance of a generic attribute is included a minimum or maximum number of times per CityGML feature. Unlike the predefined CityGML objects, the concrete layout and occurrence of generic objects and attributes cannot be validated by an XML parser. This may reduce semantic interoperability.
- Naming conflicts of generic attributes or objects can occur, if the CityGML instance documents should be augmented by specific information from different applications simultaneously.
- There is only a limited number of predefined data types that can be used for generic attributes. Also the structure of generic objects might not be appropriate to represent more complex objects.

If application specific information are well-structured, it is desirable to represent them in a systematic way, i. e. by the definition of an extra formal schema based on the CityGML schema definitions. Such an XML schema is called a CityGML *Application Domain Extension* (ADE). It allows to validate instance documents both against the extended CityGML and the ADE schema and therefore helps to maintain semantic and syntactic interoperability between different systems working in the same application field. In order to prevent naming conflicts, every ADE has to be defined within its own namespace which must differ from the namespaces associated with the CityGML modules. An ADE schema may extend one or more CityGML module schemas. The relevant CityGML module schemas have to be imported by the ADE schema.

The ADE concept defines a special way of extending existing CityGML feature types which allows to use different ADEs within the same instance document simultaneously (see below). For example, the specification of ADEs can be useful in the following application fields: cultural heritage (extension of abstract class *_CityObject* e.g. by time period information and monument protection status); representation of subsurface objects (tunnel, underpass) or city lighting (light sources like street lamps and house lights); real estate management (economic parameters of the CityGML features; inclusion of attributes defined for real estate assets as defined by OSCRE); utility networks (as topographic features); additional building properties as defined by the U.S. national building information model standard (NBIMS).

10.13.1 Technical principle of ADEs

Each ADE is specified by its own XML schema file. The target namespace is provided by the information community who specifies the CityGML ADE. This is typically not the OGC or the SIG 3D. The namespace should be in the control of this information community and must be given as a previously unused and globally

unique URI. This URI will be used in CityGML ADE instance documents to distinguish extensions from CityGML base elements. As the URI refers to the information community it also denotes the originator of the employed ADE.

The ADE's XML schema file must be available (or accessible on the Internet) to everybody creating and parsing CityGML instance documents including these ADE specific augmentations.

An ADE XML schema can define various extensions to CityGML. However, all extensions shall belong to one of the two following categories:

1. New feature types are defined within the ADE namespace and are based on CityGML abstract or concrete classes. In general, this mechanism follows the same principles as the definition of application schemas for GML. This means, that new feature types have to be derived from existing (here: CityGML) feature types. For example, new feature types could be defined by classes derived from the abstract classes like *_CityObject* or *_AbstractBuilding* or the concrete class *CityFurniture*. The new feature types then automatically inherit all properties (i.e. attributes) and associations (i.e. relations) from the respective CityGML superclasses.

2. Existing CityGML feature types are extended by application specific properties (in the ADE namespace). These properties may have simple or complex data types. Also geometries or embedded features (feature properties) are possible. The latter can also be used to model relations to other features.

   In this case, extension of the CityGML feature type is not being realised by the inheritance mechanism of XML schema. Instead, every CityGML feature type provides a "hook" in its XML schema definition, that allows to attach additional properties to it by ADEs. This "hook" is implemented as a GML property of the form "_GenericApplicationPropertyOf<*Featuretypename*>" where <*Featuretypename*> is equal to the name of the feature type definition in which it is included. The datatype for these kinds of properties is always "xsd:anyType" from the XSD namespace. The minimum occurrence of the "_GenericApplicationPropertyOf<*Featuretypename*>" is 0 and the maximum occurrence unbounded. This means, that the CityGML schema allows that every CityGML feature may have an arbitrary number of additional properties with arbitrary XML content with the name "_GenericApplicationPropertyOf<*Featuretypename*>". For example, the last property in the definition of the CityGML feature type *LandUse* is the element *_GenericApplicationPropertyOfLandUse* (cf. chapter 10.10.1).

   Such properties are called "hooks" to attach application specific properties, because they are used as the head of a substitution group by ADEs. Whenever an ADE wants to add an extra property to an existing CityGML feature type, it should declare the respective element with the appropriate datatype within the ADE namespace. In the element declaration this element shall be explicitly assigned to the substitution group defined by the corresponding "_GenericApplicationPropertyOf<*Featuretypename*>" in the corresponding CityGML module namespace. An example is given in the following subsection.

   By following this concept, it is possible to specify different ADEs for different information communities. Every ADE may add their specific properties to the same CityGML feature type as they all can belong to the same substitution group. This allows to have CityGML instance documents where CityGML features contain additional information from different ADEs simultaneously.

Please note that usage of ADEs introduces an extra level of complexity as data files may contain mixed information (features, properties) from different namespaces, not only from the GML and CityGML module namespaces. However, extended CityGML instance documents are quite easy to handle by applications that are not "schema-aware", i.e. applications that do not parse and interpret GML application schemas in a generic way. These applications can simply skip anything from a CityGML instance document that is not from a CityGML module or GML namespace. Thus, a building is still represented by the <*bldg:Building*> element with the standard CityGML properties, but with possibly some extra properties from different namespaces. Also features from a different namespace than those declared by CityGML modules or GML could be skipped (e.g. by a viewer application).

10.13.2 Example ADE

In this section, the ADE mechanism is illustrated by a short example, which deals with the application of virtual 3D city models to generate noise pollution maps. In our example, two extensions of CityGML are required for

this task: buildings have to be extended to represent a “noise reflection correction” value and the number of inhabitants. As a new feature type noise barriers have to be defined which also have a “noise reflection correction” value.

The XSD schema which has to be defined to implement this model declares a new namespace for the noise extension (*http://www.citygml.org/ade/noise_de/2.0*). Additionally, the namespaces of the extended CityGML modules are declared (for corresponding prefixes see chapter 4.3 and chapter 7), and the respective schema definition files are imported. The XML schema adds the elements *buildingReflectionCorrection* and *buildingHabitants*, both being members of the substitution group *bldg:_GenericApplicationPropertyOfAbstractBuilding* which is defined by the CityGML *Building* module. Thus, both elements may be used as child elements of CityGML building features. Noise barriers are represented as *NoiseCityFurnitureSegment* elements. The corresponding type *NoiseCityFurnitureSegmentType* is defined as subtype of the CityGML abstract type *core:AbstractCityObjectType* provided by the *CityGML Core* module, applying the usual subtyping mechanism of XML and XSD. A further element *noiseCityFurnitureSegmentProperty* is added as a member of the substitution group *frn:_GenericApplicationPropertyOfCityFurniture*. By this means, noise barriers may be modelled as child elements of CityGML city furniture objects.

The XSD file for this example CityGML Noise ADE is given by the following excerpt (the complete CityGML Noise ADE is given in Annex H):

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns="http://www.citygml.org/ade/noise_de/2.0" xmlns:gml="http://www.opengis.net/gml"
  xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xmlns:frn="http://www.opengis.net/citygml/cityfurniture/2.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"targetNamespace="http://www.citygml.org/ade/noise_de/2.0"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/2.0"
              schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/transportation/2.0"
              schemaLocation="http://schemas.opengis.net/citygml/transportation/2.0/transportation.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/building/2.0"
              schemaLocation="http://schemas.opengis.net/citygml/building/2.0/building.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/cityfurniture/2.0"
              schemaLocation="http://schemas.opengis.net/citygml/cityfurniture/2.0/cityFurniture.xsd"/>
  …
  <xsd:element name="buildingReflection" type="xsd:string"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xsd:element name="buildingHabitants" type="xsd:positiveInteger"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  …
  <xsd:element name="noiseCityFurnitureSegmentProperty" type="NoiseCityFurnitureSegmentPropertyType"
               substitutionGroup="frn:_GenericApplicationPropertyOfCityFurniture"/>
  <!-- ========================================================================================== -->
  <xsd:complexType name="NoiseCityFurnitureSegmentPropertyType">
    <xsd:sequence minOccurs="0">
      <xsd:element ref="NoiseCityFurnitureSegment" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xsd:complexType>
  <!-- ========================================================================================== -->
  <xsd:complexType name="NoiseCityFurnitureSegmentType">
    <xsd:complexContent>
      <xsd:extension base="core:AbstractCityObjectType">
        <xsd:sequence>
          <xsd:element name="type" type="NoiseCityFurnitureSegmentTypeType" minOccurs="0"/>
          <xsd:element name="reflection" type="xsd:string" minOccurs="0"/>
          <xsd:element name="reflectionCorrection" type="gml:MeasureType" minOccurs="0"/>
          <xsd:element name="height" type="gml:LengthType" minOccurs="0"/>
          <xsd:element name="distance" type="gml:LengthType" minOccurs="0"/>
          <xsd:element name="lod0BaseLine" type="gml:CurvePropertyType"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- ========================================================================================== -->
  <xsd:element name="NoiseCityFurnitureSegment" type="NoiseCityFurnitureSegmentType" substitutionGroup="core:_CityObject"/>  …
</xsd:schema>
```

An example for a feature collection in a corresponding instance document is depicted below. Two CityGML buildings contain application specific properties distinguished from CityGML properties by the namespace prefix *noise*. The other properties, function and geometry, are defined by corresponding CityGML modules. In addition to the buildings, a noise barrier as child of a city furniture element is included in the feature collection. Please note, that the order of the child elements in the sequence is not arbitrary: the child elements defined by an ADE subschema have to occur after the child elements defined by CityGML modules. There is, however, no specific order of the ADE properties.

```
…
<core:cityObjectMember>
  <bldg:Building gml:id="aa">
    <bldg:function>1004</bldg:function>
    <bldg:lod1Solid> ... </bldg:lod1Solid>
    <noise:buildingHabitants>14</noise:buildingHabitants>
    <noise:buildingReflectionCorrection uom="dB">4.123</noise:buildingReflectionCorrection>
  </bldg:Building>
</core:cityObjectMember>
<core:cityObjectMember>
  <bldg:Building gml:id="aaa">
    <bldg:function>1004</bldg:function>
    <bldg:lod1Solid> ... </bldg:lod1Solid>
    <noise:buildingReflectionCorrection uom="dB">3.123</noise:buildingReflectionCorrection>
    <noise:buildingHabitants>6</noise:buildingHabitants>
  </bldg:Building>
</core:cityObjectMember>
<core:cityObjectMember>
  <frn:CityFurniture gml:id="CFUR_0815">
    <frn:function>1520</frn:function>
    <frn:lod1Geometry> … </frn:lod1Geometry>
    <noise:noiseCityFurnitureSegmentProperty>
      <noise:NoiseCityFurnitureSegment gml:id="CFRS_0815">
        <noise:type>1</noise:type>
        <noise:reflection>absorbierende Lärmschutzwand</noise:reflection>
        <noise:reflectionCorrection uom="dB">4.123</noise:reflectionCorrection>
        <noise:height uom="m">7.123</noise:height>
        <noise:distance uom="m">21.123</noise:distance>
        <noise:lod0BaseLine>
          <gml:LineString srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783" srsDimension="3">
            <gml:coordinates decimal="." cs="," ts=" ">5707335,2524175,188 5707338,2524181,188 5707330,2524185,188
              5707327,2524179,188</gml:coordinates>
          </gml:LineString>
        </noise:lod0BaseLine>
      </noise:NoiseCityFurnitureSegment>
    </noise:noiseCityFurnitureSegmentProperty>
  </frn:CityFurniture>
</core:cityObjectMember>
…
```

10.14 Code lists

CityGML feature types often include attributes whose values can be enumerated in a list of discrete values. An example is the attribute *roof type* of a building, whose attribute values typically are saddle back roof, hip roof, semi-hip roof, flat roof, pent roof, or tent roof. If such an attribute is typed as string, misspellings or different names for the same notion obstruct interoperability.

If the list of values is fixed, the allowed attribute values are specified in and enforced by the CityGML schema using an *enumeration* as attribute type. Attributes of an enumerated type may only take values from the predefined list. Examples for such attributes are *relativeToTerrain* and *relativeToWater* of the abstract base class *core:_CityObject* (*CityGML Core* module, cf. chapter 10.1) as well as *wrapMode* of the abstract class *app:_Texture* (*Appearance* module, cf. chapter 9.4).

In case a fixed enumeration of possible attribute values is not suitable, the attribute type is specified as *gml:CodeType* and the allowed attribute values can be provided in a *code list* which is specified outside the CityGML schema. Examples for such attributes are *class*, *function*, and *usage* which are available for almost all CityGML feature types. A code list contains coded attribute values which hinder misspellings and ensure that the same code is used for the same notion or concept. If a code list is provided for an attribute of type *gml:CodeType*, then any conformant attribute shall only take values from the given code list. This allows applications to validate the attribute values and thus facilitates semantic and syntactic interoperability. The optional *codeSpace* attribute declared for *gml:CodeType* is used to associate an attribute with a code list. If a *codeSpace* is present, then its value shall be a persistent URI identifying the code list. If no *codeSpace* is given, then the attribute value can only be interpreted as a simple text token and validation requires additional knowledge.

The governance of code lists is decoupled from the governance of the CityGML schema and specification. Accordingly, code lists can be specified by any organisation or information community according to their information needs. There shall be one authority per *codeSpace* and hence per code list who is in charge of the contents and the maintenance of the code list. As a result, the code list values are managed outside the CityGML schema. Thus, in contrast to a fixed *enumeration* enforced by the CityGML schema, changes to a code list do not require a revision of the CityGML schema and specification.

The contents of code lists may substantially vary for different countries (e.g., due to national law or regulations) and for different information communities. For this reason, this International standard does not specify normative code lists for any of the attributes of type *gml:CodeType*. However, Annex C provides non-normative code lists for selected attributes which are proposed and maintained by the SIG 3D. These code lists can be directly referenced in CityGML instance documents and serve as an example for the definition of code lists. The code lists given in Annex C comprise the non-normative code lists which are included in the previous version 1.0 of this International standard in order to ensure backwards compatibility.

It is recommended that code lists are implemented as *simple dictionaries* following the *GML 3.1.1 Simple Dictionary Profile* (cf. Whiteside 2005). An example for a code list implemented as *simple dictionary* is given below. It shows an excerpt of the code list proposed by the SIG 3D for the attribute *roofType* of the class *_AbstractBuilding* (*Building* module, cf. chapter 10.3).

```
<gml:Dictionary xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/gml
  http://schemas.opengis.net/gml/3.1.1/profiles/SimpleDictionary/1.0.0/gmlSimpleDictionaryProfile.xsd"
  gml:id="roofType">
  <gml:name>roofType</gml:name>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id357">
      <gml:description>flat roof</gml:description>
      <gml:name>1000</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id358">
      <gml:description>monopitch roof</gml:description>
      <gml:name>1010</gml:name>
    </gml:Definition>
```



```
  </gml:dictionaryEntry>
  …
</gml:Dictionary>
```

Listing 2: Example of a code list implemented as simple dictionary following the *GML 3.1.1 Simple Dictionary Profile.*

In the simple dictionary concept, the code list itself is represented by a *gml:Dictionary* element. The allowed attribute values are listed as *gml:Definition* entries contained in the *gml:Dictionary*. For each definition entry, the coded attribute value is specified by the *gml:name* subelement. Any attribute referencing this code list in a CityGML instance document may only take values which are specified by a *gml:name* element of one of the definition entries. If the attribute value is not specified by one of the definition entries, then the attribute value is invalid. The *gml:description* subelement of a definition entry provides an additional textual description for the coded attribute value. This description can be used, for example, as human readable substitute for the coded attribute value.

The following excerpt of a CityGML instance document illustrates the usage of the code list mechanism. The document contains a *bldg:Building* object whose *roofType* value is taken from the code list shown in Listing 2. The *codeSpace* attribute of the *roofType* element identifies the code list through the globally unique URL *http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml* which is managed and maintained by the SIG 3D. According to this code list, the coded attribute value *1000* denotes a *flat roof* for this building.

```
<bldg:Building>
  <bldg:roofType
      codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml">1000</bldg:roofType>
  …
</bldg:Building>
```


Annex A
(normative)

XML Schema definition

A.1 CityGML Core module

The *CityGML Core* module is defined within the XML Schema definition file *cityGMLBase.xsd*. The target namespace http://www.opengis.net/citygml/2.0 is associated with the core module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/2.0" xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
             schemaLocation="http://docs.oasis-open.org/election/external/xAL.xsd"/>
  <xs:complexType name="CityModelType">
    <xs:annotation>
      <xs:documentation>Type describing the "root" element of any city model file. It is a collection whose members are restricted
        to be features of a city model. All features are included as cityObjectMember. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureCollectionType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfCityModel" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="CityModel" type="CityModelType" substitutionGroup="gml:_FeatureCollection"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfCityModel" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:element name="cityObjectMember" type="gml:FeaturePropertyType" substitutionGroup="gml:featureMember"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="AbstractCityObjectType" abstract="true">
    <xs:annotation>
      <xs:documentation>Type describing the abstract superclass of most CityGML features. Its purpose is to provide a creation and
        a termination date as well as a reference to corresponding objects in other information systems. A generalization relation
        may be used to relate features, which represent the same real-world object in different Levels-of-Detail, i.e. a feature
        and its generalized counterpart(s). The direction of this relation is from the feature to the corresponding generalized
        feature.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
          <xs:element name="creationDate" type="xs:date" minOccurs="0"/>
          <xs:element name="terminationDate" type="xs:date" minOccurs="0"/>
          <xs:element name="externalReference" type="ExternalReferenceType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="generalizesTo" type="GeneralizationRelationType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="relativeToTerrain" type="RelativeToTerrainType" minOccurs="0"/>
          <xs:element name="relativeToWater" type="RelativeToWaterType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfCityObject" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_CityObject" type="AbstractCityObjectType" abstract="true" substitutionGroup="gml:_Feature"/>
```

```
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfCityObject" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="AbstractSiteType" abstract="true">
  <xs:annotation>
    <xs:documentation>Type describing the abstract superclass for buildings, facilities, etc. Future extensions of CityGML like
      bridges and tunnels would be modelled as subclasses of _Site. As subclass of _CityObject, a _Site inherits all attributes
      and relations, in particular an id, names, external references, and generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractCityObjectType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfSite" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_Site" type="AbstractSiteType" abstract="true" substitutionGroup="_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfSite" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="GeneralizationRelationType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of a _CityObject to its corresponding _CityObject in higher LOD, i.e. to the
      _CityObjects representing the same real world object in higher LOD. The GeneralizationRelationType element must either
      carry a reference to a _CityObject object or contain a _CityObject object inline, but neither both nor none.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="_CityObject"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:complexType name="ExternalReferenceType">
  <xs:annotation>
    <xs:documentation>Type describing the reference to an corresponding object in an other information system, for example in
      the german cadastre ALKIS, the german topographic information system or ATKIS, or the OS MasterMap. The reference consists
      of the name of the external information system, represented by an URI, and the reference of the external object, given
      either by a string or by an URI. If the informationSystem element is missing in the ExternalReference, the
      ExternalObjectReference must be an URI, which contains an indication of the informationSystem.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="informationSystem" type="xs:anyURI" minOccurs="0"/>
    <xs:element name="externalObject" type="ExternalObjectReferenceType"/>
  </xs:sequence>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:complexType name="ExternalObjectReferenceType">
  <xs:choice>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="uri" type="xs:anyURI"/>
  </xs:choice>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:simpleType name="RelativeToTerrainType">
  <xs:annotation>
    <xs:documentation>Specifies the spatial relation of a CityObject realativ to terrain in a qualitative way. The values of
      this type are defined in the XML file RelativeToTerrainType.xml, according to the dictionary concept of
      GML3.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="entirelyAboveTerrain"/>
    <xs:enumeration value="substantiallyAboveTerrain"/>
    <xs:enumeration value="substantiallyAboveAndBelowTerrain"/>
    <xs:enumeration value="substantiallyBelowTerrain"/>
    <xs:enumeration value="entirelyBelowTerrain"/>
  </xs:restriction>
</xs:simpleType>
<!-- ========================================================================================== -->
<xs:simpleType name="RelativeToWaterType">
  <xs:annotation>
    <xs:documentation>Specifies the spatial relation of a CityObject realativ to the water surface in a qualitative way. The
      values of this type are defined in the XML file RelativeToTerrainType.xml, according to the dictionary concept of
      GML3.</xs:documentation>
```

```
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="entirelyAboveWaterSurface"/>
      <xs:enumeration value="substantiallyAboveWaterSurface"/>
      <xs:enumeration value="substantiallyAboveAndBelowWaterSurface"/>
      <xs:enumeration value="substantiallyBelowWaterSurface"/>
      <xs:enumeration value="entirelyBelowWaterSurface"/>
      <xs:enumeration value="temporarilyAboveAndBelowWaterSurface"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- ========================================================================================== -->
  <xs:complexType name="AddressPropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of an _CityObject to its addresses. The AddressPropertyType element must either carry
        a reference to an Address object or contain an Address object inline, but neither both nor none. </xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="Address"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="AddressType">
    <xs:annotation>
      <xs:documentation>Type for addresses. It references the xAL address standard issued by the OASIS consortium. Please note,
        that addresses are modelled as GML features. Every address can be assigned zero or more 2D or 3D point geometries (one
        gml:MultiPoint geometry) locating the entrance(s). </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
          <xs:element name="xalAddress" type="xalAddressPropertyType"/>
          <xs:element name="multiPoint" type="gml:MultiPointPropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfAddress" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Address" type="AddressType" substitutionGroup="gml:_Feature"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfAddress" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="xalAddressPropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of an Address feature to the xAL address element.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element ref="xAL:AddressDetails"/>
    </xs:sequence>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="ImplicitGeometryType">
    <xs:annotation>
      <xs:documentation> Type for the implicit representation of a geometry. An implicit geometry is a geometric object, where the
        shape is stored only once as a prototypical geometry, e.g. a tree or other vegetation object, a traffic light or a traffic
        sign. This prototypic geometry object is re-used or referenced many times, wherever the corresponding feature occurs in
        the 3D city model. Each occurrence is represented by a link to the prototypic shape geometry (in a local cartesian
        coordinate system), by a transforma-tion matrix that is multiplied with each 3D coordinate tuple of the prototype, and by
        an anchor point denoting the base point of the object in the world coordinate reference system. In order to determine the
        absolute coordinates of an implicit geometry, the anchor point coordinates have to be added to the matrix multiplication
        results. The transformation matrix accounts for the intended rotation, scaling, and local translation of the prototype. It
        is a 4x4 matrix that is multiplied with the prototype coordinates using homogeneous coordinates, i.e. (x,y,z,1). This way
        even a projection might be modelled by the transformation matrix. The concept of implicit geometries is an enhancement of
        the geometry model of GML3. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:AbstractGMLType">
        <xs:sequence>
          <xs:element name="mimeType" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="transformationMatrix" type="TransformationMatrix4x4Type" minOccurs="0"/>
          <xs:element name="libraryObject" type="xs:anyURI" minOccurs="0"/>
          <xs:element name="relativeGMLGeometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="referencePoint" type="gml:PointPropertyType"/>
        </xs:sequence>
      </xs:extension>
```

```
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="ImplicitGeometry" type="ImplicitGeometryType" substitutionGroup="gml:_GML"/>
<!-- ========================================================================================== -->
<xs:complexType name="ImplicitRepresentationPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of a _CityObject to its implicit geometry representation, which is a representation
      of a geometry by referencing a prototype and transforming it to its real position in space. The
      ImplicitRepresentationPropertyType element must either carry a reference to a ImplicitGeometry object or contain a
      ImplicitGeometry object inline, but neither both nor none. </xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="ImplicitGeometry"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:simpleType name="doubleBetween0and1">
  <xs:annotation>
    <xs:documentation>Type for values, which are greater or equal than 0 and less or equal than 1. Used for color encoding, for
      example. </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:double">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="1"/>
  </xs:restriction>
</xs:simpleType>
<!-- ========================================================================================== -->
<xs:simpleType name="doubleBetween0and1List">
  <xs:annotation>
    <xs:documentation>List for double values, which are greater or equal than 0 and less or equal than 1. Used for color
      encoding, for example. </xs:documentation>
  </xs:annotation>
  <xs:list itemType="doubleBetween0and1"/>
</xs:simpleType>
<!-- ========================================================================================== -->
<xs:simpleType name="TransformationMatrix4x4Type">
  <xs:annotation>
    <xs:documentation>Used for implicit geometries. The Transformation matrix is a 4 by 4 matrix, thus it must be a list with 16
      items. The order the matrix element are represented is row-major, i. e. the first 4 elements represent the first row, the
      fifth to the eight element the second row,... </xs:documentation>
  </xs:annotation>
  <xs:restriction base="gml:doubleList">
    <xs:length value="16"/>
  </xs:restriction>
</xs:simpleType>
<!-- ========================================================================================== -->
<xs:simpleType name="TransformationMatrix2x2Type">
  <xs:annotation>
    <xs:documentation>Used for georeferencing. The Transformation matrix is a 2 by 2 matrix, thus it must be a list with 4
      items. The order the matrix element are represented is row-major, i. e. the first 2 elements represent the first row, the
      fifth to the eight element the second row,... </xs:documentation>
  </xs:annotation>
  <xs:restriction base="gml:doubleList">
    <xs:length value="4"/>
  </xs:restriction>
</xs:simpleType>
<!-- ========================================================================================== -->
<xs:simpleType name="TransformationMatrix3x4Type">
  <xs:annotation>
    <xs:documentation>Used for texture parameterization. The Transformation matrix is a 3 by 4 matrix, thus it must be a list
      with 12 items. The order the matrix element are represented is row-major, i. e. the first 4 elements represent the first
      row, the fifth to the eight element the second row,... </xs:documentation>
  </xs:annotation>
  <xs:restriction base="gml:doubleList">
    <xs:length value="12"/>
  </xs:restriction>
</xs:simpleType>
<!-- ========================================================================================== -->
<xs:simpleType name="integerBetween0and4">
  <xs:annotation>
    <xs:documentation>Type for integer values, which are greater or equal than 0 and less or equal than 4. Used for encoding of
      the LOD number. </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:integer">
```

```
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="4"/>
      </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

A.2 Appearance module

The CityGML *Appearance* module is defined within the XML Schema definition file *appearance.xsd*. The target namespace http://www.opengis.net/citygml/appearance/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/appearance/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/appearance/2.0" elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xs:complexType name="AppearanceType">
    <xs:annotation>
      <xs:documentation> Named container for all surface data (texture/material). All appearances of the same name ("theme")
        within a CityGML file are considered a group. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
          <xs:element name="theme" type="xs:string" minOccurs="0"/>
          <xs:element name="surfaceDataMember" type="SurfaceDataPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfAppearance" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Appearance" type="AppearanceType" substitutionGroup="gml:_Feature"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfAppearance" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="AppearancePropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of a _CityObject to its appearances. The AppearancePropertyType element must either
        carry a reference to a Appearance object or contain a Appearance object inline, but neither both nor
        none.</xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="Appearance"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="appearanceMember" type="gml:FeaturePropertyType" substitutionGroup="gml:featureMember"/>
  <!-- ========================================================================================== -->
  <xs:element name="appearance" type="AppearancePropertyType" substitutionGroup="core:_GenericApplicationPropertyOfCityObject"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="AbstractSurfaceDataType" abstract="true">
    <xs:annotation>
      <xs:documentation>Base class for textures and material. Contains only isFront-flag.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
          <xs:element name="isFront" type="xs:boolean" default="true" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfSurfaceData" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_SurfaceData" type="AbstractSurfaceDataType" abstract="true" substitutionGroup="gml:_Feature"/>
  <!-- ========================================================================================== -->
```



```
<xs:element name="_GenericApplicationPropertyOfSurfaceData" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<xs:complexType name="SurfaceDataPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of an Appearance to its surface data. The SurfaceDataPropertyType element must either
      carry a reference to a _SurfaceData object or contain a _SurfaceData object inline, but neither both nor
      none.</xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="_SurfaceData" minOccurs="0"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:complexType name="AbstractTextureType" abstract="true">
  <xs:annotation>
    <xs:documentation>Base class for textures. "imageURI" can contain any valid URI from references to a local file to
      preformatted web service requests. The linking to geometry and texture parameterization is provided by derived
      classes.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractSurfaceDataType">
      <xs:sequence>
        <xs:element name="imageURI" type="xs:anyURI"/>
        <xs:element name="mimeType" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="textureType" type="TextureTypeType" minOccurs="0"/>
        <xs:element name="wrapMode" type="WrapModeType" minOccurs="0"/>
        <xs:element name="borderColor" type="ColorPlusOpacity" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfTexture" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="_Texture" type="AbstractTextureType" abstract="true" substitutionGroup="_SurfaceData"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfTexture" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<xs:simpleType name="WrapModeType">
  <xs:annotation>
    <xs:documentation>Fill mode for a texture. "wrap" repeats the texture, "clamp" extends the edges of the texture, and
      "border" fills all undefined areas with "borderColor"</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="wrap"/>
    <xs:enumeration value="mirror"/>
    <xs:enumeration value="clamp"/>
    <xs:enumeration value="border"/>
  </xs:restriction>
</xs:simpleType>
<!-- ========================================================================================= -->
<xs:complexType name="ParameterizedTextureType">
  <xs:annotation>
    <xs:documentation>Specialization for standard 2D textures. "target" provides the linking to surface geometry. Only
      gml:MultiSurface and decendants of gml:AbstractSurfaceType are valid targets. As property of the link, a texture
      parameterization either as set of texture coordinates or transformation matrix is given. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractTextureType">
      <xs:sequence>
        <xs:element name="target" type="TextureAssociationType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfParameterizedTexture" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="ParameterizedTexture" type="ParameterizedTextureType" substitutionGroup="_Texture"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfParameterizedTexture" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<xs:complexType name="GeoreferencedTextureType">
  <xs:annotation>
    <xs:documentation>Specialization for georeferenced textures, i.e. textures using a planimetric projection. Such textures
      contain an implicit parameterization (either stored within the image file, in an acompanying world file, or using the
```

```
        "referencePoint" and "orientation"-elements). A georeference provided by "referencePoint" and "orientation" always takes
        precedence. The search order for an external georeference is determined by the boolean flag preferWorldFile. If this flag
        is set to true (its default value), a world file is looked for first and only if it is not found the georeference from the
        image data is used. If preferWorldFile is false, the world file is used only if no georeference from the image data is
        available. The "boundedBy"-property should contain the bounding box of the projected image data. Since a georeferenced
        texture has a unique parameterization, "target" only provides links to surface geometry without any additional texture
        parameterization. Only gml:MultiSurface or decendants of gml:AbstractSurfaceType are valid targets.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractTextureType">
        <xs:sequence>
          <xs:element name="preferWorldFile" type="xs:boolean" default="true" minOccurs="0"/>
          <xs:element name="referencePoint" type="gml:PointPropertyType" minOccurs="0"/>
          <xs:element name="orientation" type="core:TransformationMatrix2x2Type" minOccurs="0"/>
          <xs:element name="target" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfGeoreferencedTexture" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="GeoreferencedTexture" type="GeoreferencedTextureType" substitutionGroup="_Texture"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfGeoreferencedTexture" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="TextureAssociationType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of a texture to a surface, that is augmented by a TextureParameterization object. The
        TextureAssociationType element must either carry a reference to a _TextureParameterization object or contain a
        _TextureParameterization object inline, but neither both nor none.</xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="_TextureParameterization"/>
    </xs:sequence>
    <xs:attribute name="uri" type="xs:anyURI" use="required"/>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="AbstractTextureParameterizationType" abstract="true">
    <xs:annotation>
      <xs:documentation>Base class for augmenting a link "texture->surface" with texture parameterization. Subclasses of this
        class define concrete parameterizations. Currently, texture coordinates and texture coordinate generation using a
        transformation matrix are available. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:AbstractGMLType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfTextureParameterization" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_TextureParameterization" type="AbstractTextureParameterizationType" abstract="true"
    substitutionGroup="gml:_GML"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfTextureParameterization" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="TexCoordListType">
    <xs:annotation>
      <xs:documentation>Texture parameterization using texture coordinates: Each gml:LinearRing that is part of the surface
        requires a separate "textureCoordinates"-entry with 2 doubles per ring vertex. The "ring"- attribute provides the gml:id
        of the target LinearRing. It is prohibited to link texture coordinates to any other object type than LinearRing. Thus,
        surfaces not consisting of LinearRings cannot be textured this way. Use transformation matrices (see below) or
        georeferenced textures instead. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractTextureParameterizationType">
        <xs:sequence>
          <xs:element name="textureCoordinates" maxOccurs="unbounded">
            <xs:complexType>
              <xs:simpleContent>
                <xs:extension base="gml:doubleList">
                  <xs:attribute name="ring" type="xs:anyURI" use="required"/>
                </xs:extension>
              </xs:simpleContent>
```

```
            </xs:complexType>
          </xs:element>
          <xs:element ref="_GenericApplicationPropertyOfTexCoordList" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="TexCoordList" type="TexCoordListType" substitutionGroup="_TextureParameterization"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfTexCoordList" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="TexCoordGenType">
    <xs:annotation>
      <xs:documentation>Texture parameterization using a transformation matrix. The transformation matrix "worldToTexture" can be
        used to derive texture coordinates from an object's location. This 3x4 matrix T computes the coordinates (s,t) from a
        homogeneous world position p as (s,t) = (s'/q', t'/q') with (s', t', q') = T*p. Thus, perspective projections can be
        specified. The SRS can be specified using standard attributes. If an object is given in a different reference system, it
        is transformed to the SRS before applying the transformation. A transformation matrix can be used for whole surfaces. It
        is not required to specify it per LinearRing. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractTextureParameterizationType">
        <xs:sequence>
          <xs:element name="worldToTexture">
            <xs:complexType>
              <xs:simpleContent>
                <xs:extension base="core:TransformationMatrix3x4Type">
                  <xs:attributeGroup ref="gml:SRSReferenceGroup"/>
                </xs:extension>
              </xs:simpleContent>
            </xs:complexType>
          </xs:element>
          <xs:element ref="_GenericApplicationPropertyOfTexCoordGen" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="TexCoordGen" type="TexCoordGenType" substitutionGroup="_TextureParameterization"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfTexCoordGen" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="X3DMaterialType">
    <xs:annotation>
      <xs:documentation>Class for defining constant surface properties. It is based on X3D's material definition. In addition,
        "isSmooth" provides a hint for value interpolation. The link to surface geometry is established via the "target"-property.
        Only gml:MultiSurface or decendants of gml:AbstractSurfaceType are valid targets. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractSurfaceDataType">
        <xs:sequence>
          <xs:element name="ambientIntensity" type="core:doubleBetween0and1" default="0.2" minOccurs="0"/>
          <xs:element name="diffuseColor" type="Color" default="0.8 0.8 0.8" minOccurs="0"/>
          <xs:element name="emissiveColor" type="Color" default="0.0 0.0 0.0" minOccurs="0"/>
          <xs:element name="specularColor" type="Color" default="1.0 1.0 1.0" minOccurs="0"/>
          <xs:element name="shininess" type="core:doubleBetween0and1" default="0.2" minOccurs="0"/>
          <xs:element name="transparency" type="core:doubleBetween0and1" default="0.0" minOccurs="0"/>
          <xs:element name="isSmooth" type="xs:boolean" default="false" minOccurs="0"/>
          <xs:element name="target" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfX3DMaterial" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="X3DMaterial" type="X3DMaterialType" substitutionGroup="_SurfaceData"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfX3DMaterial" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:simpleType name="TextureTypeType">
    <xs:annotation>
      <xs:documentation>Textures can be qualified by the attribute textureType. The textureType differentiates between textures,
        which are specific for a certain object and are only used for that object (specific), and prototypic textures being
        typical for that kind of object and are used many times for all objects of that kind (typical). A typical texture may be
        replaced by a specific, if available. Textures may also be classified as unknown. </xs:documentation>
```

```
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="specific"/>
      <xs:enumeration value="typical"/>
      <xs:enumeration value="unknown"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- ================================================================================== -->
  <xs:simpleType name="Color">
    <xs:annotation>
      <xs:documentation>List of three values (red, green, blue), separated by spaces. The values must be in the range between zero
        and one. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="core:doubleBetween0and1List">
      <xs:length value="3"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- ================================================================================== -->
  <xs:simpleType name="ColorPlusOpacity">
    <xs:annotation>
      <xs:documentation>List of three or four values (red, green, blue, opacity), separated by spaces. The values must be in the
        range between zero and one. If no opacity is given, it is assumed as 1.0.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="core:doubleBetween0and1List">
      <xs:minLength value="3"/>
      <xs:maxLength value="4"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

A.3 Bridge module

The CityGML *Bridge* module is defined within the XML Schema definition file *bridge.xsd*. The target namespace http://www.opengis.net/citygml/bridge/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/bridge/2.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:core="http://www.opengis.net/citygml/2.0" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/bridge/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xs:complexType name="AbstractBridgeType" abstract="true">
    <xs:annotation>
      <xs:documentation>Type describing the thematic and geometric attributes and the associations of bridges. It is an abstract
        type, only its subclasses Bridge and BridgePart can be instantiated. An _AbstractBridge may consist of BridgeParts, which
        are again _AbstractBridges by inheritance. Thus an aggregation hierarchy between _AbstractBridges of arbitrary depth may
        be specified. In such an hierarchy, top elements are Bridges, while all other elements are BridgeParts. Each element of
        such a hierarchy may have all attributes and geometries of _AbstractBridges. It must, however, be assured that no
        inconsistencies occur.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractSiteType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="yearOfConstruction" type="xs:gYear" minOccurs="0"/>
          <xs:element name="yearOfDemolition" type="xs:gYear" minOccurs="0"/>
          <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
          <xs:element name="lod1Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod2Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod2MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="outerBridgeConstruction" type="BridgeConstructionElementPropertyType" minOccurs="0"
                      maxOccurs="unbounded"/>
          <xs:element name="outerBridgeInstallation" type="BridgeInstallationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="interiorBridgeInstallation" type="IntBridgeInstallationPropertyType" minOccurs="0"
                      maxOccurs="unbounded"/>
          <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod3Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod3MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="interiorBridgeRoom" type="InteriorBridgeRoomPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="consistsOfBridgePart" type="BridgePartPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="address" type="core:AddressPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfAbstractBridge" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_AbstractBridge" type="AbstractBridgeType" abstract="true" substitutionGroup="core:_Site"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfAbstractBridge" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
```



```
<xs:complexType name="BridgeType">
  <xs:complexContent>
    <xs:extension base="AbstractBridgeType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfBridge" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Bridge" type="BridgeType" substitutionGroup="_AbstractBridge"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBridge" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="BridgePartType">
  <xs:complexContent>
    <xs:extension base="AbstractBridgeType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfBridgePart" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="BridgePart" type="BridgePartType" substitutionGroup="_AbstractBridge"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBridgePart" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="BridgePartPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of an _AbstractBridge to its bridge parts. The BridgePartPropertyType element must
      either carry a reference to a BridgePart object or contain a BridgePart object inline, but neither both nor
      none.</xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="BridgePart"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:complexType name="BridgeInstallationType">
  <xs:annotation>
    <xs:documentation>A BridgeInstallation is a part of a Bridge which has not the significance of a BridgePart. In contrast to
      BridgeConstructionElements, a BridgeInstallation is not essential from a structural point of view. Thus, it may be removed
      without the bridge collapsing. Examples are stairs, antennas, railways, etc. As subclass of _CityObject, a
      BridgeInstallation inherits all attributes and relations, in particular an id, names, external references, generic
      attributes and generalization relations.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfBridgeInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="BridgeInstallation" type="BridgeInstallationType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBridgeInstallation" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="BridgeInstallationPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of an _AbstractBridge to its bridge installations. The BridgeInstallationPropertyType
      element must either carry a reference to a BridgeInstallation object or contain a BridgeInstallation object inline, but
      neither both nor none.</xs:documentation>
```

```
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="BridgeInstallation"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="IntBridgeInstallationType">
    <xs:annotation>
      <xs:documentation>An IntBridgeInstallation is an interior part of a Bridge which has a specific function or semantic
        meaning. Examples are interior stairs, railings, radiators or pipes. As subclass of _CityObject, an IntBridgeInstallation
        inherits all attributes and relations, in particular an id, names, external references, generic attributes and
        generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfIntBridgeInstallation" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="IntBridgeInstallation" type="IntBridgeInstallationType" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfIntBridgeInstallation" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="IntBridgeInstallationPropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of an _AbstractBridge to its interior bridge installations. The
        IntBridgeInstallationPropertyType element must either carry a reference to a IntBridgeInstallation object or contain a
        IntBridgeInstallation object inline, but neither both nor none.</xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="IntBridgeInstallation"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="BridgeConstructionElementType">
    <xs:annotation>
      <xs:documentation>A BridgeConstructionElement is a part of a Bridge which has not the significance of a BridgePart. In
        contrast to BridgeInstallation, a BridgeConstructionElement is essential from a structural point of view. Examples are
        pylons, anchorages, etc. As subclass of _CityObject, a BridgeInstallation inherits all attributes and relations, in
        particular an id, names, external references, generic attributes and generalization relations.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:annotation>
            <xs:documentation> The name will be represented by gml:name (inherited from _GML) The lodXMultiSurface must be used,
              if the geometry of a building is just a collection of surfaces bounding a solid, but not a topologically clean solid
              boundary necessary for GML3 solid boundaries. </xs:documentation>
          </xs:annotation>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
```

```
            <xs:element ref="_GenericApplicationPropertyOfBridgeConstructionElement" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:element name="BridgeConstructionElement" type="BridgeConstructionElementType" substitutionGroup="core:_CityObject"/>
    <!-- ========================================================================================== -->
    <xs:element name="_GenericApplicationPropertyOfBridgeConstructionElement" type="xs:anyType"/>
    <!-- ========================================================================================== -->
    <xs:complexType name="BridgeConstructionElementPropertyType">
      <xs:annotation>
        <xs:documentation>Denotes the relation of an _AbstractBridge to its bridge construction elements. The
          BridgeConstructionElementPropertyType element must either carry a reference to a BridgeConstructionElement object or
          contain a BridgeConstructionElement object inline, but neither both nor none.</xs:documentation>
      </xs:annotation>
      <xs:sequence minOccurs="0">
        <xs:element ref="BridgeConstructionElement"/>
      </xs:sequence>
      <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:complexType name="AbstractBoundarySurfaceType" abstract="true">
      <xs:annotation>
        <xs:documentation>A BoundarySurface is a thematic object which classifies surfaces bounding an _AbstractBridge,
          BridgeInstallation, IntBuildingInstallation, BridgeConstructionElement, and BridgeRoom. The geometry of a BoundarySurface
          is given by MultiSurfaces. As it is a subclass of _CityObject, it inherits all atributes and relations, in particular the
          external references, the generic attributes, and the generalization relations. </xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="core:AbstractCityObjectType">
          <xs:sequence>
            <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="opening" type="OpeningPropertyType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="_GenericApplicationPropertyOfBoundarySurface" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:element name="_BoundarySurface" type="AbstractBoundarySurfaceType" abstract="true" substitutionGroup="core:_CityObject"/>
    <!-- ========================================================================================== -->
    <xs:element name="_GenericApplicationPropertyOfBoundarySurface" type="xs:anyType" abstract="true"/>
    <!-- ========================================================================================== -->
    <xs:complexType name="RoofSurfaceType">
      <xs:complexContent>
        <xs:extension base="AbstractBoundarySurfaceType">
          <xs:sequence>
            <xs:element ref="_GenericApplicationPropertyOfRoofSurface" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:element name="RoofSurface" type="RoofSurfaceType" substitutionGroup="_BoundarySurface"/>
    <!-- ========================================================================================== -->
    <xs:element name="_GenericApplicationPropertyOfRoofSurface" type="xs:anyType" abstract="true"/>
    <!-- ========================================================================================== -->
    <xs:complexType name="WallSurfaceType">
      <xs:complexContent>
        <xs:extension base="AbstractBoundarySurfaceType">
          <xs:sequence>
            <xs:element ref="_GenericApplicationPropertyOfWallSurface" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:element name="WallSurface" type="WallSurfaceType" substitutionGroup="_BoundarySurface"/>
    <!-- ========================================================================================== -->
    <xs:element name="_GenericApplicationPropertyOfWallSurface" type="xs:anyType" abstract="true"/>
    <!-- ========================================================================================== -->
    <xs:complexType name="GroundSurfaceType">
      <xs:complexContent>
```

```
      <xs:extension base="AbstractBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="GroundSurface" type="GroundSurfaceType" substitutionGroup="_BoundarySurface"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfGroundSurface" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="ClosureSurfaceType">
    <xs:complexContent>
      <xs:extension base="AbstractBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="ClosureSurface" type="ClosureSurfaceType" substitutionGroup="_BoundarySurface"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfClosureSurface" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="OuterFloorSurfaceType">
    <xs:complexContent>
      <xs:extension base="AbstractBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfOuterFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="OuterFloorSurface" type="OuterFloorSurfaceType" substitutionGroup="_BoundarySurface"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfOuterFloorSurface" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="OuterCeilingSurfaceType">
    <xs:complexContent>
      <xs:extension base="AbstractBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfOuterCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="OuterCeilingSurface" type="OuterCeilingSurfaceType" substitutionGroup="_BoundarySurface"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfOuterCeilingSurface" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="FloorSurfaceType">
    <xs:complexContent>
      <xs:extension base="AbstractBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="FloorSurface" type="FloorSurfaceType" substitutionGroup="_BoundarySurface"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfFloorSurface" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="InteriorWallSurfaceType">
    <xs:complexContent>
      <xs:extension base="AbstractBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfInteriorWallSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
```

```
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="InteriorWallSurface" type="InteriorWallSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfInteriorWallSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="CeilingSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="CeilingSurface" type="CeilingSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfCeilingSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="BoundarySurfacePropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of an _AbstractBridge to its bounding thematic surfaces (walls, roofs, ..). The
      BoundarySurfacePropertyType element must either carry a reference to a _BoundarySurface object or contain a
      _BoundarySurface object inline, but neither both nor none. There is no differentiation between interior surfaces bounding
      rooms and outer ones bounding bridges (one reason is, that ClosureSurface belongs to both types). It has to be made sure
      by additional integrity constraints that, e.g. an _AbstractBridge is not related to CeilingSurfaces or a room not to
      RoofSurfaces.</xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="_BoundarySurface"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:complexType name="OpeningPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of an _BondarySurface to its openings (doors, windows). The OpeningPropertyType
      element must either carry a reference to an _Opening object or contain an _Opening object inline, but neither both nor
      none.</xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="_Opening"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:complexType name="AbstractOpeningType" abstract="true">
  <xs:annotation>
    <xs:documentation> Type for openings (doors, windows) in boundary surfaces. Used in LoD3 and LoD4 only. As subclass of
      _CityObject, an _Opening inherits all attributes and relations, in particular an id, names, external references, generic
      attributes and generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfOpening" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="_Opening" type="AbstractOpeningType" abstract="true" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfOpening" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="WindowType">
  <xs:annotation>
    <xs:documentation> Type for windows in boundary surfaces. Used in LoD3 and LoD4 only . As subclass of _CityObject, a window
      inherits all attributes and relations, in particular an id, names, external references, generic attributes and
      generalization relations. </xs:documentation>
  </xs:annotation>
```

```
    <xs:complexContent>
      <xs:extension base="AbstractOpeningType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfWindow" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Window" type="WindowType" substitutionGroup="_Opening"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfWindow" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="DoorType">
    <xs:annotation>
      <xs:documentation> Type for doors in boundary surfaces. Used in LoD3 and LoD4 only . As subclass of _CityObject, a Door
        inherits all attributes and relations, in particular an id, names, external references, generic attributes and
        generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractOpeningType">
        <xs:sequence>
          <xs:element name="address" type="core:AddressPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfDoor" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Door" type="DoorType" substitutionGroup="_Opening"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfDoor" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="BridgeRoomType">
    <xs:annotation>
      <xs:documentation>A BridgeRoom is a thematic object for modelling the closed parts inside a Bridge. It has to be closed, if
        necessary by using closure surfaces. The geometry may be either a solid, or a MultiSurface if the boundary is not
        topologically clean. The BridgeRoom connectivity may be derived by detecting shared thematic openings or closure surfaces:
        two rooms are connected if both use the same opening object or the same closure surface. The thematic surfaces bounding a
        BridgeRoom are referenced by the boundedBy property. As subclass of _CityObject, a BridgeRoom inherits all attributes and
        relations, in particular an id, names, external references, generic attributes and generalization relations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="interiorFurniture" type="InteriorFurniturePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="bridgeRoomInstallation" type="IntBridgeInstallationPropertyType" minOccurs="0"
                      maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfBridgeRoom" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="BridgeRoom" type="BridgeRoomType" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfBridgeRoom" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="BridgeFurnitureType">
    <xs:annotation>
      <xs:documentation>Type for bridge furnitures. As subclass of _CityObject, a BridgeFurniture inherits all attributes and
        relations, in particular an id, names, external references, generic attributes and generalization relations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
```

```
            <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
            <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
            <xs:element ref="_GenericApplicationPropertyOfBridgeFurniture" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:element name="BridgeFurniture" type="BridgeFurnitureType" substitutionGroup="core:_CityObject"/>
    <!-- ========================================================================================== -->
    <xs:element name="_GenericApplicationPropertyOfBridgeFurniture" type="xs:anyType" abstract="true"/>
    <!-- ========================================================================================== -->
    <xs:complexType name="InteriorBridgeRoomPropertyType">
      <xs:annotation>
        <xs:documentation>Denotes the relation of an _AbstractBridge to its rooms. The InteriorBridgeRoomPropertyType element must
          either carry a reference to an BridgeRoom object or contain an BridgeRoom object inline, but neither both nor
          none.</xs:documentation>
      </xs:annotation>
      <xs:sequence minOccurs="0">
        <xs:element ref="BridgeRoom"/>
      </xs:sequence>
      <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:complexType name="InteriorFurniturePropertyType">
      <xs:annotation>
        <xs:documentation>Denotes the relation of a BridgeRoom to its interior bridge furniture. The
          InteriorBridgeFurniturePropertyType element must either carry a reference to an BridgeFurniture object or contain an
          BridgeFurniture object inline, but neither both nor none.</xs:documentation>
      </xs:annotation>
      <xs:sequence minOccurs="0">
        <xs:element ref="BridgeFurniture"/>
      </xs:sequence>
      <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
    </xs:complexType>
  </xs:schema>
```

A.4 Building module

The CityGML *Building* module is defined within the XML Schema definition file *building.xsd*. The target namespace http://www.opengis.net/citygml/building/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/building/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/building/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xs:complexType name="AbstractBuildingType" abstract="true">
    <xs:annotation>
      <xs:documentation>Type describing the thematic and geometric attributes and the associations of buildings. It is an abstract
        type, only its subclasses Building and BuildingPart can be instantiated. An _AbstractBuilding may consist of
        BuildingParts, which are again _AbstractBuildings by inheritance. Thus an aggregation hierarchy between _AbstractBuildings
        of arbitrary depth may be specified. In such an hierarchy, top elements are Buildings, while all other elements are
        BuildingParts. Each element of such a hierarchy may have all attributes and geometries of _AbstractBuildings. It must,
        however, be assured than no inconsistencies occur (for example, if the geometry of a Building does not correspond to the
        geometries of its parts, or if the roof type of a Building is saddle roof, while its parts have an hip roof). As subclass
        of _CityObject, an _AbstractBuilding inherits all attributes and relations, in particular an id, names, external
        references, and generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractSiteType">
        <xs:sequence>
          <xs:annotation>
            <xs:documentation> The name will be represented by gml:name (inherited from _GML) . list order for
              storeyHeightsAboveground: first floor, second floor,... list order for storeyHeightsBelowground: first floor below
              ground, second floor below ground,... The lodXMultiSurface must be used, if the geometry of a building is just a
              collection of surfaces bounding a solid, but not a topologically clean solid boundary necessary for GML3 solid
              boundaries. </xs:documentation>
          </xs:annotation>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="yearOfConstruction" type="xs:gYear" minOccurs="0"/>
          <xs:element name="yearOfDemolition" type="xs:gYear" minOccurs="0"/>
          <xs:element name="roofType" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="measuredHeight" type="gml:LengthType" minOccurs="0"/>
          <xs:element name="storeysAboveGround" type="xs:nonNegativeInteger" minOccurs="0"/>
          <xs:element name="storeysBelowGround" type="xs:nonNegativeInteger" minOccurs="0"/>
          <xs:element name="storeyHeightsAboveGround" type="gml:MeasureOrNullListType" minOccurs="0"/>
          <xs:element name="storeyHeightsBelowGround" type="gml:MeasureOrNullListType" minOccurs="0"/>
          <xs:element name="lod0FootPrint" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod0RoofEdge" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod1Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod2Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod2MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="outerBuildingInstallation" type="BuildingInstallationPropertyType" minOccurs="0"
                      maxOccurs="unbounded"/>
          <xs:element name="interiorBuildingInstallation" type="IntBuildingInstallationPropertyType" minOccurs="0"
                      maxOccurs="unbounded"/>
          <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod3Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod3MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
```



```
            <xs:element name="lod4MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="interiorRoom" type="InteriorRoomPropertyType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="consistsOfBuildingPart" type="BuildingPartPropertyType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="address" type="core:AddressPropertyType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="_GenericApplicationPropertyOfAbstractBuilding" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:element name="_AbstractBuilding" type="AbstractBuildingType" abstract="true" substitutionGroup="core:_Site"/>
    <!-- ========================================================================================== -->
    <xs:element name="_GenericApplicationPropertyOfAbstractBuilding" type="xs:anyType" abstract="true"/>
    <!-- ========================================================================================== -->
    <xs:complexType name="BuildingType">
      <xs:complexContent>
        <xs:extension base="AbstractBuildingType">
          <xs:sequence>
            <xs:element ref="_GenericApplicationPropertyOfBuilding" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:element name="Building" type="BuildingType" substitutionGroup="_AbstractBuilding"/>
    <!-- ========================================================================================== -->
    <xs:element name="_GenericApplicationPropertyOfBuilding" type="xs:anyType" abstract="true"/>
    <!-- ========================================================================================== -->
    <xs:complexType name="BuildingPartType">
      <xs:complexContent>
        <xs:extension base="AbstractBuildingType">
          <xs:sequence>
            <xs:element ref="_GenericApplicationPropertyOfBuildingPart" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:element name="BuildingPart" type="BuildingPartType" substitutionGroup="_AbstractBuilding"/>
    <!-- ========================================================================================== -->
    <xs:element name="_GenericApplicationPropertyOfBuildingPart" type="xs:anyType" abstract="true"/>
    <!-- ========================================================================================== -->
    <xs:complexType name="BuildingPartPropertyType">
      <xs:annotation>
        <xs:documentation>Denotes the relation of an _AbstractBuilding to its building parts. The BuildingPartPropertyType element
          must either carry a reference to a BuildingPart object or contain a BuildingPart object inline, but neither both nor
          none.</xs:documentation>
      </xs:annotation>
      <xs:sequence minOccurs="0">
        <xs:element ref="BuildingPart"/>
      </xs:sequence>
      <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:complexType name="BuildingInstallationType">
      <xs:annotation>
        <xs:documentation>A BuildingInstallation is a part of a Building which has not the significance of a BuildingPart. Examples
          are stairs, antennas, balconies or small roofs. As subclass of _CityObject, a BuildingInstallation inherits all attributes
          and relations, in particular an id, names, external references, and generalization relations. </xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="core:AbstractCityObjectType">
          <xs:sequence>
            <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
            <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
            <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
            <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
            <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
            <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
            <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
            <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="_GenericApplicationPropertyOfBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
```

```
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="BuildingInstallation" type="BuildingInstallationType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfBuildingInstallation" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="BuildingInstallationPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of an _AbstractBuilding to its building installations. The
      BuildingInstallationPropertyType element must either carry a reference to a BuildingInstallation object or contain a
      BuildingInstallation object inline, but neither both nor none. </xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="BuildingInstallation"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:complexType name="IntBuildingInstallationType">
  <xs:annotation>
    <xs:documentation>An IntBuildingInstallation is an interior part of a Building which has a specific function or semantical
      meaning. Examples are interior stairs, railings, radiators or pipes. As subclass of _CityObject, a
      nIntBuildingInstallation inherits all attributes and relations, in particular an id, names, external references, and
      generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfIntBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="IntBuildingInstallation" type="IntBuildingInstallationType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfIntBuildingInstallation" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="IntBuildingInstallationPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of an _AbstractBuilding to its interior building installations. The
      IntBuildingInstallationPropertyType element must either carry a reference to a IntBuildingInstallation object or contain a
      IntBuildingInstallation object inline, but neither both nor none. </xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="IntBuildingInstallation"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:complexType name="AbstractBoundarySurfaceType" abstract="true">
  <xs:annotation>
    <xs:documentation>A BoundarySurface is a thematic object which classifies surfaces bounding an _AbstractBuilding, Room,
      BuildingInstallation, and IntBuildingInstallation. The geometry of a BoundarySurface is given by MultiSurfaces. As it is a
      subclass of _CityObject, it inherits all atributes and relations, in particular the external references, and the
      generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="opening" type="OpeningPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfBoundarySurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
```

```
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="_BoundarySurface" type="AbstractBoundarySurfaceType" abstract="true" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfBoundarySurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<xs:complexType name="RoofSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfRoofSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="RoofSurface" type="RoofSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfRoofSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<xs:complexType name="WallSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWallSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="WallSurface" type="WallSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfWallSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<xs:complexType name="GroundSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="GroundSurface" type="GroundSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfGroundSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<xs:complexType name="ClosureSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="ClosureSurface" type="ClosureSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfClosureSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<xs:complexType name="FloorSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="FloorSurface" type="FloorSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfFloorSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
```

```
<xs:complexType name="OuterFloorSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfOuterFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="OuterFloorSurface" type="OuterFloorSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfOuterFloorSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<xs:complexType name="InteriorWallSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfInteriorWallSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="InteriorWallSurface" type="InteriorWallSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfInteriorWallSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<xs:complexType name="CeilingSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="CeilingSurface" type="CeilingSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfCeilingSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<xs:complexType name="OuterCeilingSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfOuterCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:element name="OuterCeilingSurface" type="OuterCeilingSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================= -->
<xs:element name="_GenericApplicationPropertyOfOuterCeilingSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================= -->
<xs:complexType name="BoundarySurfacePropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of an _AbstractBuilding to its bounding thematic surfaces (walls, roofs, ..). The
      BoundarySurfacePropertyType element must either carry a reference to a _BoundarySurface object or contain a
      _BoundarySurface object inline, but neither both nor none. There is no differentiation between interior surfaces bounding
      rooms and outer ones bounding buildings (one reason is, that ClosureSurface belongs to both types). It has to be made sure
      by additional integrity constraints that, e.g. an _AbstractBuilding is not related to CeilingSurfaces or a room not to
      RoofSurfaces. </xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="_BoundarySurface"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:complexType name="OpeningPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of an _BondarySurface to its openings (doors, windows). The OpeningPropertyType
      element must either carry a reference to an _Opening object or contain an _Opening object inline, but neither both nor
      none. </xs:documentation>
```

```
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="_Opening"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="AbstractOpeningType" abstract="true">
    <xs:annotation>
      <xs:documentation> Type for openings (doors, windows) in boundary surfaces. Used in LOD3 and LOD4 only. As subclass of
        _CityObject, an _Opening inherits all attributes and relations, in particular an id, names, external references, and
        generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfOpening" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_Opening" type="AbstractOpeningType" abstract="true" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfOpening" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="WindowType">
    <xs:annotation>
      <xs:documentation> Type for windows in boundary surfaces. Used in LOD3 and LOD4 only . As subclass of _CityObject, a window
        inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractOpeningType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfWindow" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Window" type="WindowType" substitutionGroup="_Opening"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfWindow" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="DoorType">
    <xs:annotation>
      <xs:documentation> Type for doors in boundary surfaces. Used in LOD3 and LOD4 only . As subclass of _CityObject, a Door
        inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractOpeningType">
        <xs:sequence>
          <xs:element name="address" type="core:AddressPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfDoor" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Door" type="DoorType" substitutionGroup="_Opening"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfDoor" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="RoomType">
    <xs:annotation>
      <xs:documentation>A Room is a thematic object for modelling the closed parts inside a building. It has to be closed, if
        necessary by using closure surfaces. The geometry may be either a solid, or a MultiSurface if the boundary is not
        topologically clean. The room connectivity may be derived by detecting shared thematic openings or closure surfaces: two
        rooms are connected if both use the same opening object or the same closure surface. The thematic surfaces bounding a room
        are referenced by the boundedBy property. As subclass of _CityObject, a Room inherits all attributes and relations, in
```

```
            particular an id, names, external references, and generalization relations. </xs:documentation>
        </xs:annotation>
        <xs:complexContent>
          <xs:extension base="core:AbstractCityObjectType">
            <xs:sequence>
              <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
              <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
              <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
              <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
              <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
              <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
              <xs:element name="interiorFurniture" type="InteriorFurniturePropertyType" minOccurs="0" maxOccurs="unbounded"/>
              <xs:element name="roomInstallation" type="IntBuildingInstallationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
              <xs:element ref="_GenericApplicationPropertyOfRoom" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      <!-- ========================================================================================== -->
      <xs:element name="Room" type="RoomType" substitutionGroup="core:_CityObject"/>
      <!-- ========================================================================================== -->
      <xs:element name="_GenericApplicationPropertyOfRoom" type="xs:anyType" abstract="true"/>
      <!-- ========================================================================================== -->
      <xs:complexType name="BuildingFurnitureType">
        <xs:annotation>
          <xs:documentation>Type for building furnitures. As subclass of _CityObject, a BuildingFurniture inherits all attributes and
            relations, in particular an id, names, external references, and generalization relations. </xs:documentation>
        </xs:annotation>
        <xs:complexContent>
          <xs:extension base="core:AbstractCityObjectType">
            <xs:sequence>
              <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
              <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
              <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
              <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
              <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
              <xs:element ref="_GenericApplicationPropertyOfBuildingFurniture" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      <!-- ========================================================================================== -->
      <xs:element name="BuildingFurniture" type="BuildingFurnitureType" substitutionGroup="core:_CityObject"/>
      <!-- ========================================================================================== -->
      <xs:element name="_GenericApplicationPropertyOfBuildingFurniture" type="xs:anyType" abstract="true"/>
      <!-- ========================================================================================== -->
      <xs:complexType name="InteriorRoomPropertyType">
        <xs:annotation>
          <xs:documentation>Denotes the relation of an _AbstractBuilding to its rooms. The InteriorRoomPropertyType element must
            either carry a reference to a Room object or contain a Room object inline, but neither both nor none. </xs:documentation>
        </xs:annotation>
        <xs:sequence minOccurs="0">
          <xs:element ref="Room"/>
        </xs:sequence>
        <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
      </xs:complexType>
      <!-- ========================================================================================== -->
      <xs:complexType name="InteriorFurniturePropertyType">
        <xs:annotation>
          <xs:documentation>Denotes the relation of a Room to its interior furnitures (movable). The InteriorFurniturePropertyType
            element must either carry a reference to a BuildingFurniture object or contain a BuildingFurniture object inline, but
            neither both nor none. </xs:documentation>
        </xs:annotation>
        <xs:sequence minOccurs="0">
          <xs:element ref="BuildingFurniture"/>
        </xs:sequence>
        <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
      </xs:complexType>
    </xs:schema>
```

A.5 CityFurniture module

The CityGML *CityFurniture* module is defined within the XML Schema definition file *cityFurniture.xsd*. The target namespace http://www.opengis.net/citygml/cityfurniture/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/cityfurniture/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/cityfurniture/2.0" elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xs:complexType name="CityFurnitureType">
    <xs:annotation>
      <xs:documentation>Type describing city furnitures, like traffic lights, benches, ... As subclass of _CityObject, a
        CityFurniture inherits all attributes and relations, in particular an id, names, external references, and generalization
        relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfCityFurniture" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="CityFurniture" type="CityFurnitureType" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfCityFurniture" type="xs:anyType" abstract="true"/>
</xs:schema>
```

A.6 CityObjectGroup module

The CityGML *CityObjectGroup* module is defined within the XML Schema definition file *cityObjectGroup.xsd*. The target namespace http://www.opengis.net/citygml/cityobjectgroup/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/cityobjectgroup/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/cityobjectgroup/2.0" elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="CityObjectGroupType">
    <xs:annotation>
      <xs:documentation> A group may be used to aggregate arbitrary CityObjects according to some user-defined criteria. Examples
        for groups are the buildings in a specific region, the result of a query, or objects put together for visualization
        purposes. Each group has a name (inherited from AbstractGMLType), functions (e.g., building group), a class and zero or
        more usages. A geometry may optionally be attached to a group, if the geometry of the whole group differs from the
        geometry of the parts. Each member of a group may be qualified by a role name, reflecting the role each CityObject plays
        in the context of the group. As subclass of _CityObject, a CityObjectGroup inherits all attributes and relations, in
        particular an id, names, external references, and generalization relations. As CityObjectGroup itself is a CityObject, it
        may also be contained by another group. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="groupMember" type="CityObjectGroupMemberType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="parent" type="CityObjectGroupParentType" minOccurs="0"/>
          <xs:element name="geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfCityObjectGroup" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="CityObjectGroup" type="CityObjectGroupType" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfCityObjectGroup" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="CityObjectGroupMemberType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of a CityObjectGroup to its members, which are _CityObjects. The
        CityObjectGroupMemberType element must either carry a reference to a _CityObject object or contain a _CityObject object
        inline, but neither both nor none. </xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="core:_CityObject"/>
    </xs:sequence>
    <xs:attribute name="role" type="xs:string"/>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="CityObjectGroupParentType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of a CityObjectGroup to its parent, which is a CityObject. The
        CityObjectGroupParentType element must either carry a reference to a _CityObject object or contain a _CityObject object
        inline, but neither both nor none. The parent association allows for modelling of a generic hierarchical grouping concept.
        Named aggregations of components (CityObjects) can be added to specific CityObjects considered as the parent object. The
        parent association links to the aggregate, while the parts are given by the group members. </xs:documentation>
```



```
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="core:_CityObject"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
</xs:schema>
```

A.7 Generics module

The CityGML *Generics* module is defined within the XML Schema definition file *generics.xsd*. The target namespace http://www.opengis.net/citygml/generics/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/generics/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/generics/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xs:complexType name="GenericCityObjectType">
    <xs:annotation>
      <xs:documentation>Generic (user defined) city objects may be used to model features which are not covered explicitly by the
        CityGML schema. Generic objects must be used with care; they shall only be used if there is no appropiate thematic class
        available in the overall CityGML schema. Oherwise, problems concerning semantic interoperability may arise. As subclass of
        _CityObject, a generic city object inherits all attributes and relations, in particular an id, names, external references,
        and generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod0Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod0TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod0ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="GenericCityObject" type="GenericCityObjectType" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="AbstractGenericAttributeType" abstract="true">
    <xs:annotation>
      <xs:documentation> Generic (user defined) attributes may be used to represent attributes which are not covered explicitly by
        the CityGML schema. Generic attributes must be used with care; they shall only be used if there is no appropiate attribute
        available in the overall CityGML schema. Oherwise, problems concerning semantic interoperability may arise. A generic
        attribute has a name and a value, which has further subclasses (IntAttrribute, StringAttribute, ...). </xs:documentation>
    </xs:annotation>
    <xs:sequence/>
    <xs:attribute name="name" type="xs:string" use="required"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_genericAttribute" type="AbstractGenericAttributeType" abstract="true"
    substitutionGroup="core:_GenericApplicationPropertyOfCityObject"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="StringAttributeType">
```

```
    <xs:annotation>
      <xs:documentation/>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractGenericAttributeType">
        <xs:sequence>
          <xs:element name="value" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================= -->
  <xs:element name="stringAttribute" type="StringAttributeType" substitutionGroup="_genericAttribute"/>
  <!-- ========================================================================================= -->
  <xs:complexType name="IntAttributeType">
    <xs:annotation>
      <xs:documentation/>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractGenericAttributeType">
        <xs:sequence>
          <xs:element name="value" type="xs:integer"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================= -->
  <xs:element name="intAttribute" type="IntAttributeType" substitutionGroup="_genericAttribute"/>
  <!-- ========================================================================================= -->
  <xs:complexType name="DoubleAttributeType">
    <xs:annotation>
      <xs:documentation/>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractGenericAttributeType">
        <xs:sequence>
          <xs:element name="value" type="xs:double"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================= -->
  <xs:element name="doubleAttribute" type="DoubleAttributeType" substitutionGroup="_genericAttribute"/>
  <!-- ========================================================================================= -->
  <xs:complexType name="DateAttributeType">
    <xs:annotation>
      <xs:documentation/>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractGenericAttributeType">
        <xs:sequence>
          <xs:element name="value" type="xs:date"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================= -->
  <xs:element name="dateAttribute" type="DateAttributeType" substitutionGroup="_genericAttribute"/>
  <!-- ========================================================================================= -->
  <xs:complexType name="UriAttributeType">
    <xs:annotation>
      <xs:documentation/>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractGenericAttributeType">
        <xs:sequence>
          <xs:element name="value" type="xs:anyURI"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================= -->
  <xs:element name="uriAttribute" type="UriAttributeType" substitutionGroup="_genericAttribute"/>
  <!-- ========================================================================================= -->
  <xs:complexType name="MeasureAttributeType">
    <xs:annotation>
```

```
        <xs:documentation/>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="AbstractGenericAttributeType">
          <xs:sequence>
            <xs:element name="value" type="gml:MeasureType"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================= -->
    <xs:element name="measureAttribute" type="MeasureAttributeType" substitutionGroup="_genericAttribute"/>
    <!-- ========================================================================================= -->
    <xs:complexType name="GenericAttributeSetType">
      <xs:annotation>
        <xs:documentation>Set of generic attributes with an optional codeSpace. If the codeSpace attribute is present, then its
          value should identify an authority for the set, such as the organisation or community who defined its content. The generic
          attribute set may contain arbitrary generic attributes.</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="AbstractGenericAttributeType">
          <xs:sequence>
            <xs:element ref="_genericAttribute" minOccurs="1" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="codeSpace" type="xs:anyURI" use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================= -->
    <xs:element name="genericAttributeSet" type="GenericAttributeSetType" substitutionGroup="_genericAttribute"/>
</xs:schema>
```

A.8 LandUse module

The CityGML *LandUse* module is defined within the XML Schema definition file *landUse.xsd*. The target namespace http://www.opengis.net/citygml/landuse/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/landuse/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/landuse/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xs:complexType name="LandUseType">
    <xs:annotation>
      <xs:documentation>Type describing the class for Land Use in all LOD. LandUse objects describe areas of the earth's surface
        dedicated to a specific land use. The geometry must consist of 3-D surfaces. As subclass of _CityObject, a LandUse
        inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod0MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfLandUse" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="LandUse" type="LandUseType" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfLandUse" type="xs:anyType" abstract="true"/>
</xs:schema>
```

A.9 Relief module

The CityGML *Relief* module is defined within the XML Schema definition file *relief.xsd*. The target namespace http://www.opengis.net/citygml/relief/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/relief/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/relief/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xs:complexType name="ReliefFeatureType">
    <xs:annotation>
      <xs:documentation>Type describing the features of the Digital Terrain Model. As subclass of _CityObject, a ReliefFeature
        inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="lod" type="core:integerBetween0and4"/>
          <xs:element name="reliefComponent" type="ReliefComponentPropertyType" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfReliefFeature" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="ReliefFeature" type="ReliefFeatureType" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfReliefFeature" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="AbstractReliefComponentType" abstract="true">
    <xs:annotation>
      <xs:documentation>Type describing the components of a relief feature - either a TIN, a Grid, mass points or break lines. As
        subclass of _CityObject, a ReliefComponent inherits all attributes and relations, in particular an id, names, external
        references, and generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="lod" type="core:integerBetween0and4"/>
          <xs:element name="extent" type="gml:PolygonPropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfReliefComponent" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_ReliefComponent" type="AbstractReliefComponentType" abstract="true" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfReliefComponent" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="ReliefComponentPropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of a ReliefFeature to its components. The ReliefComponentPropertyType element must
        either carry a reference to a _ReliefComponent object or contain a _ReliefComponent object inline, but neither both nor
        none. </xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="_ReliefComponent"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
```



```
</xs:complexType>
<!-- ========================================================================================== -->
<xs:complexType name="TINReliefType">
  <xs:annotation>
    <xs:documentation>Type describing the TIN component of a relief feature. As subclass of _CityObject, a TINRelief inherits
      all attributes and relations, in particular an id, names, external references, and generalization relations.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractReliefComponentType">
      <xs:sequence>
        <xs:element name="tin" type="tinPropertyType"/>
        <xs:element ref="_GenericApplicationPropertyOfTinRelief" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="TINRelief" type="TINReliefType" substitutionGroup="_ReliefComponent"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTinRelief" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="RasterReliefType">
  <xs:annotation>
    <xs:documentation>Type describing the raster component of a relief feature. As subclass of _CityObject, a RasterRelief
      inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractReliefComponentType">
      <xs:sequence>
        <xs:element name="grid" type="gridPropertyType"/>
        <xs:element ref="_GenericApplicationPropertyOfRasterRelief" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="RasterRelief" type="RasterReliefType" substitutionGroup="_ReliefComponent"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfRasterRelief" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="MassPointReliefType">
  <xs:annotation>
    <xs:documentation>Type describing the mass point component of a relief feature. As subclass of _CityObject, a MassPoint
      Relief inherits all attributes and relations, in particular an id, names, external references, and generalization
      relations. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractReliefComponentType">
      <xs:sequence>
        <xs:element name="reliefPoints" type="gml:MultiPointPropertyType"/>
        <xs:element ref="_GenericApplicationPropertyOfMassPointRelief" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="MassPointRelief" type="MassPointReliefType" substitutionGroup="_ReliefComponent"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfMassPointRelief" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="BreaklineReliefType">
  <xs:annotation>
    <xs:documentation>Type describing the break line Component of a relief feature. A break line relief consists of break lines
      or ridgeOrValleyLines. As subclass of _CityObject, a BreaklineRelief inherits all attributes and relations, in particular
      an id, names, external references, and generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractReliefComponentType">
      <xs:sequence>
        <xs:element name="ridgeOrValleyLines" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="breaklines" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfBreaklineRelief" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
```

```
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="BreaklineRelief" type="BreaklineReliefType" substitutionGroup="_ReliefComponent"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfBreaklineRelief" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="tinPropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of a TINRelief to its components. The tinPropertyType element must either carry a
        reference to a gml:TriangulatedSurface object or contain a gml:TriangulatedSurface object inline, but neither both nor
        none. </xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="gml:TriangulatedSurface"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="gridPropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of a RasterReliefType to its components. The gridPropertyType element must either
        carry a reference to a gml:RectifiedGridCoverage object or contain a gml:RectifiedGridCoverage object inline, but neither
        both nor none. </xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="gml:RectifiedGridCoverage"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Elevation" type="gml:LengthType" substitutionGroup="gml:_Object"/>
</xs:schema>
```

A.10 Transportation module

The CityGML *Transportation* module is defined within the XML Schema definition file *transportation.xsd*. The target namespace http://www.opengis.net/citygml/transportation/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/transportation/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/transportation/2.0" elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xs:complexType name="AbstractTransportationObjectType" abstract="true">
    <xs:annotation>
      <xs:documentation>Type describing the abstract superclass for transportation objects. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfTransportationObject" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_TransportationObject" type="AbstractTransportationObjectType" abstract="true"
    substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfTransportationObject" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="TransportationComplexType">
    <xs:annotation>
      <xs:documentation>Type describing transportation complexes, which are aggregated features, e.g. roads, which consist of
        parts (traffic areas, e.g. pedestrian path, and auxiliary traffic areas). As subclass of _CityObject, a
        TransportationComplex inherits all attributes and relations, in particular an id, names, external references, and
        generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractTransportationObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="trafficArea" type="TrafficAreaPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="auxiliaryTrafficArea" type="AuxiliaryTrafficAreaPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod0Network" type="gml:GeometricComplexPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfTransportationComplex" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="TransportationComplex" type="TransportationComplexType" substitutionGroup="_TransportationObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfTransportationComplex" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="TrafficAreaType">
    <xs:annotation>
      <xs:documentation>Type describing the class for traffic Areas. Traffic areas are the surfaces where traffic actually takes
        place. As subclass of _CityObject, a TrafficArea inherits all attributes and relations, in particular an id, names,
```



```
        external references, and generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractTransportationObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="surfaceMaterial" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfTrafficArea" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="TrafficArea" type="TrafficAreaType" substitutionGroup="_TransportationObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfTrafficArea" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="AuxiliaryTrafficAreaType">
    <xs:annotation>
      <xs:documentation>Type describing the class for auxiliary traffic Areas. These are the surfaces where no traffic actually
        takes place, but which belong to a transportation object. Examples are kerbstones, road markings and grass stripes. As
        subclass of _CityObject, an AuxiliaryTrafficArea inherits all attributes and relations, in particular an id, names,
        external references, and generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractTransportationObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="surfaceMaterial" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfAuxiliaryTrafficArea" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="AuxiliaryTrafficArea" type="AuxiliaryTrafficAreaType" substitutionGroup="_TransportationObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfAuxiliaryTrafficArea" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="TrafficAreaPropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of TransportationComplex to its parts, which are traffic areas. The
        TrafficAreaPropertyType element must either carry a reference to a TrafficArea object or contain a TrafficArea object
        inline, but neither both nor none. </xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="TrafficArea"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="AuxiliaryTrafficAreaPropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of TransportationComplex to its parts, which are auxiliary traffic areas. The
        TrafficAreaPropertyType element must either carry a reference to a TrafficArea object or contain a TrafficArea object
        inline, but neither both nor none. </xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="AuxiliaryTrafficArea"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="TrackType">
    <xs:annotation>
      <xs:documentation>Type describing the class for tracks. A track is a small path mainly used by pedestrians. As subclass of
```

```
        _CityObject, a Track inherits all attributes and relations, in particular an id, names, external references, and
        generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="TransportationComplexType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfTrack" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Track" type="TrackType" substitutionGroup="TransportationComplex"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfTrack" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="RoadType">
    <xs:annotation>
      <xs:documentation>Type describing the class for roads. As subclass of _CityObject, a Road inherits all attributes and
        relations, in particular an id, names, external references, and generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="TransportationComplexType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfRoad" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Road" type="RoadType" substitutionGroup="TransportationComplex"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfRoad" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="RailwayType">
    <xs:annotation>
      <xs:documentation>Type describing the class for railways. As subclass of _CityObject, a Railway inherits all attributes and
        relations, in particular an id, names, external references, and generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="TransportationComplexType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfRailway" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Railway" type="RailwayType" substitutionGroup="TransportationComplex"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfRailway" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="SquareType">
    <xs:annotation>
      <xs:documentation>Type describing the class for squares. A square is an open area commonly found in cities (like a plaza).
        As subclass of _CityObject, a Square inherits all attributes and relations, in particular an id, names, external
        references, and generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="TransportationComplexType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfSquare" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Square" type="SquareType" substitutionGroup="TransportationComplex"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfSquare" type="xs:anyType" abstract="true"/>
</xs:schema>
```

A.11 Tunnel module

The CityGML *Tunnel* module is defined within the XML Schema definition file *tunnel.xsd*. The target namespace http://www.opengis.net/citygml/tunnel/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/tunnel/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/tunnel/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xs:complexType name="AbstractTunnelType">
    <xs:annotation>
      <xs:documentation>Abstract super class of the features Tunnel and TunnelPart</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractSiteType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="yearOfConstruction" type="xs:gYear" minOccurs="0"/>
          <xs:element name="yearOfDemolition" type="xs:gYear" minOccurs="0"/>
          <xs:element name="lod1Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod2Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod2MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="outerTunnelInstallation" type="TunnelInstallationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="interiorTunnelInstallation" type="IntTunnelInstallationPropertyType" minOccurs="0"
                      maxOccurs="unbounded"/>
          <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod3Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod3MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="interiorHollowSpace" type="InteriorHollowSpacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="consistsOfTunnelPart" type="TunnelPartPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfAbstractTunnel" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_AbstractTunnel" type="AbstractTunnelType" abstract="true" substitutionGroup="core:_Site"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfAbstractTunnel" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="TunnelType">
    <xs:annotation>
      <xs:documentation>Horizontal or sloping underground or partly underground, enclosed way of some length (ISO
        6707-1)</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractTunnelType">
        <xs:sequence>
```



```
        <xs:element ref="_GenericApplicationPropertyOfTunnel" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="Tunnel" type="TunnelType" substitutionGroup="_AbstractTunnel"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTunnel" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="TunnelPartType">
  <xs:annotation>
    <xs:documentation>A Tunnel composed of structural segments differing in important geometrical or semantical properties can
      be separated into one Tunnel and additional TunnelParts. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractTunnelType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfTunnelPart" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="TunnelPart" type="TunnelPartType" substitutionGroup="_AbstractTunnel"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTunnelPart" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="TunnelPartPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of an _AbstractTunnel to its parts. The TunnelPartPropertyType element must either
      carry a reference to a TunnelPart object or contain a TunnelPart object inline, but neither both nor
      none.</xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="TunnelPart"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:complexType name="TunnelInstallationType">
  <xs:annotation>
    <xs:documentation>Immovable structural component of a Tunnel which has not the significance of a TunnelPart. As subclass of
      _CityObject, a TunnelInstallation inherits all attributes and relations, in particular an id, names, external references,
      generic attributes and generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfTunnelInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="TunnelInstallation" type="TunnelInstallationType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTunnelInstallation" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="TunnelInstallationPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of a Tunnel to its external installations. The TunnelInstallationPropertyType element
      must either carry a reference to a TunnelInstallation object or contain a TunnelInstallation object inline, but neither
      both nor none. </xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
```

```
      <xs:element ref="TunnelInstallation"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="IntTunnelInstallationType">
    <xs:annotation>
      <xs:documentation>Immovable interior structural component of a Tunnel or a HollowSpace. Examples are interior stairs,
        railings, radiators or pipes. As subclass of _CityObject, an IntTunnelInstallation inherits all attributes and relations,
        in particular an id, names, external references, and generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfIntTunnelInstallation" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="IntTunnelInstallation" type="IntTunnelInstallationType" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfIntTunnelInstallation" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="IntTunnelInstallationPropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of a Tunnel to its internal installations. The InteriorTunnelInstallationPropertyType
        element must either carry a reference to a IntTunnelInstallation object or contain a IntTunnelInstallation object inline,
        but neither both nor none. </xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="IntTunnelInstallation"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="BoundarySurfacePropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of an tunnel or hollow space to its bounding thematic surfaces (walls, roofs, ..).
        There is no differentiation between interior surfaces bounding hollow spaces and outer ones bounding tunnels (one reason
        is, that ClosureSurface belongs to both types). It has to be made sure by additional integrity constraints that, e.g. a
        tunnel is not related to CeilingSurfaces or a room not to RoofSurfaces. </xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="_BoundarySurface"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="AbstractBoundarySurfaceType" abstract="true">
    <xs:annotation>
      <xs:documentation>Abstract super class of the features RoofSurface, WallSurface, GroundSurface, ClosureSurface,
        FloorSurface, OuterFloorSurface, CeilingSurface and OuterCeilingSurface</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="opening" type="OpeningPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfBoundarySurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_BoundarySurface" type="AbstractBoundarySurfaceType" abstract="true" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfBoundarySurface" type="xs:anyType" abstract="true"/>
```

```
<!-- ========================================================================================== -->
<xs:complexType name="RoofSurfaceType">
  <xs:annotation>
    <xs:documentation>Construction that separates the interior part of the Tunnel from the ambient medium ( water, air, ..) from
      above.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfRoofSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="RoofSurface" type="RoofSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfRoofSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="WallSurfaceType">
  <xs:annotation>
    <xs:documentation>Mainly vertical construction that separates the interior part of the Tunnel from the ambient medium (rock,
      earth, water, air, ..)</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWallSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="WallSurface" type="WallSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfWallSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="GroundSurfaceType">
  <xs:annotation>
    <xs:documentation>Horizontal construction that separates the interior part of the Tunnel from the ambient medium (rock,
      earth, water, air, ..) from below on the lowest level. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="GroundSurface" type="GroundSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfGroundSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="ClosureSurfaceType">
  <xs:annotation>
    <xs:documentation>Virtual surface which can be used to define the volume of geometric objects being not totally bounded by
      real surfaces (e. g. the entrance of an open Tunnel).</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="ClosureSurface" type="ClosureSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfClosureSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="FloorSurfaceType">
  <xs:annotation>
    <xs:documentation>Mostly horizontal construction that bounds a HollowSpace from below. </xs:documentation>
```

```
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="FloorSurface" type="FloorSurfaceType" substitutionGroup="_BoundarySurface"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfFloorSurface" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="OuterFloorSurfaceType">
    <xs:annotation>
      <xs:documentation>Horizontal construction that separates the interior part of the Tunnel from the ambient medium (rock,
        earth, water, air, ..) from below.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfOuterFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="OuterFloorSurface" type="OuterFloorSurfaceType" substitutionGroup="_BoundarySurface"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfOuterFloorSurface" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="InteriorWallSurfaceType">
    <xs:annotation>
      <xs:documentation>Mostly vertical construction that bounds a HollowSpace.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfInteriorWallSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="InteriorWallSurface" type="InteriorWallSurfaceType" substitutionGroup="_BoundarySurface"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfInteriorWallSurface" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="CeilingSurfaceType">
    <xs:annotation>
      <xs:documentation>Mostly horizontal construction that bounds a HollowSpace from above.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="CeilingSurface" type="CeilingSurfaceType" substitutionGroup="_BoundarySurface"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfCeilingSurface" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="OuterCeilingSurfaceType">
    <xs:annotation>
      <xs:documentation>Mainly horizontal construction that separates the interior part of the Tunnel from the ambient medium
        (rock, earth, ..) from above.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfOuterCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
```

```
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="OuterCeilingSurface" type="OuterCeilingSurfaceType" substitutionGroup="_BoundarySurface"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfOuterCeilingSurface" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="HollowSpaceType">
  <xs:annotation>
    <xs:documentation>Area or volume within a Tunnel bounded actually or theoretically </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="interiorFurniture" type="InteriorFurniturePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="hollowSpaceInstallation" type="IntTunnelInstallationPropertyType" minOccurs="0"
                    maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfHollowSpace" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="HollowSpace" type="HollowSpaceType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfHollowSpace" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="InteriorHollowSpacePropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of a Tunnel to its internal holllow spaces. The InteriorHollowSpacePropertyType
      element must either carry a reference to a HollowSpace object or contain a HollowSpace object inline, but neither both nor
      none. </xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="HollowSpace"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:complexType name="TunnelFurnitureType">
  <xs:annotation>
    <xs:documentation>Movable, functional objects, whether useful or ornamental, usually found in a
      HollowSpace</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfTunnelFurniture" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================== -->
<xs:element name="TunnelFurniture" type="TunnelFurnitureType" substitutionGroup="core:_CityObject"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfTunnelFurniture" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:complexType name="InteriorFurniturePropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of a hollow space to the furnitures it contains. </xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="TunnelFurniture"/>
  </xs:sequence>
```

```
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="AbstractOpeningType" abstract="true">
    <xs:annotation>
      <xs:documentation> Type for openings (doors, windows) in boundary surfaces. Used in LOD3 and LOD4 only. As subclass of
        _CityObject, an _Opening inherits all attributes and relations, in particular an id, names, external references, and
        generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfOpening" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_Opening" type="AbstractOpeningType" abstract="true" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfOpening" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="OpeningPropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of an _BondarySurface to its openings (doors, windows). The OpeningPropertyType
        element must either carry a reference to an _Opening object or contain an _Opening object inline, but neither both nor
        none. </xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="_Opening"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="WindowType">
    <xs:annotation>
      <xs:documentation>Construction for closing an _Opening not intended for access or regress.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractOpeningType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfWindow" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Window" type="WindowType" substitutionGroup="_Opening"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfWindow" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="DoorType">
    <xs:annotation>
      <xs:documentation>Construction for closing an _Opening intended primarily for access or regress or both. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractOpeningType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfDoor" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Door" type="DoorType" substitutionGroup="_Opening"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfDoor" type="xs:anyType" abstract="true"/>
</xs:schema>
```

A.12 Vegetation module

The CityGML *Vegetation* module is defined within the XML Schema definition file *vegetation.xsd*. The target namespace http://www.opengis.net/citygml/vegetation/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/vegetation/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/vegetation/2.0" elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xs:complexType name="AbstractVegetationObjectType" abstract="true">
    <xs:annotation>
      <xs:documentation>Type describing the abstract superclass for vegetation objects. A subclass is either a
        SolitaryVegetationObject or a PlantCover. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfVegetationObject" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_VegetationObject" type="AbstractVegetationObjectType" abstract="true" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfVegetationObject" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="PlantCoverType">
    <xs:annotation>
      <xs:documentation>Type describing Plant Covers resp. Biotopes. As subclass of _CityObject, a VegetationObject inherits all
        attributes and relations, in particular an id, names, external references, and generalization relations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractVegetationObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="averageHeight" type="gml:LengthType" minOccurs="0"/>
          <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod1MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0"/>
          <xs:element name="lod2MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0"/>
          <xs:element name="lod3MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0"/>
          <xs:element name="lod4MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfPlantCover" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="PlantCover" type="PlantCoverType" substitutionGroup="_VegetationObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfPlantCover" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="SolitaryVegetationObjectType">
    <xs:annotation>
      <xs:documentation>Type describing solitary vegetation objects, e.g., trees. Its geometry is either defined explicitly by a
        GML 3 geometry with absolute coordinates, or in the case of multiple occurences of the same vegetation object, implicitly
```



```
        by a reference to a shape definition and a transformation. The shape definition may be given in an external file. As
        subclass of _CityObject, a SolitaryVegetationObject inherits all attributes and relations, in particular an id, names,
        external references, and generalization relations. </xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="AbstractVegetationObjectType">
          <xs:sequence>
            <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
            <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="species" type="gml:CodeType" minOccurs="0"/>
            <xs:element name="height" type="gml:LengthType" minOccurs="0"/>
            <xs:element name="trunkDiameter" type="gml:LengthType" minOccurs="0"/>
            <xs:element name="crownDiameter" type="gml:LengthType" minOccurs="0"/>
            <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
            <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
            <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
            <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
            <xs:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
            <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
            <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
            <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
            <xs:element ref="_GenericApplicationPropertyOfSolitaryVegetationObject" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- ========================================================================================== -->
    <xs:element name="SolitaryVegetationObject" type="SolitaryVegetationObjectType" substitutionGroup="_VegetationObject"/>
    <!-- ========================================================================================== -->
    <xs:element name="_GenericApplicationPropertyOfSolitaryVegetationObject" type="xs:anyType" abstract="true"/>
</xs:schema>
```

A.13 WaterBody module

The CityGML *WaterBody* module is defined within the XML Schema definition file *waterBody.xsd*. The target namespace http://www.opengis.net/citygml/waterbody/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/waterbody/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/waterbody/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xs:complexType name="AbstractWaterObjectType" abstract="true">
    <xs:annotation>
      <xs:documentation>Type describing the abstract superclass for water objects. As subclass of _CityObject, a _WaterObject
        inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfWaterObject" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_WaterObject" type="AbstractWaterObjectType" abstract="true" substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfWaterObject" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="WaterBodyType">
    <xs:annotation>
      <xs:documentation>Type describing Water Bodies, e.g., lakes, rivers. As subclass of _CityObject, a WaterBody inherits all
        attributes and relations, in particular an id, names, external references, and generalization relations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractWaterObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod0MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod0MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod1MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
          <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod1Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod2Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod3Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
          <xs:element name="boundedBy" type="BoundedByWaterSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfWaterBody" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="WaterBody" type="WaterBodyType" substitutionGroup="_WaterObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfWaterBody" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="BoundedByWaterSurfacePropertyType">
    <xs:annotation>
```



```
      <xs:documentation>Denotes the relation of a WaterBody to its boundary surfaces, which are of type _WaterBoundarySurface. The
        BoundedByWaterSurfacePropertyType element must either carry a reference to a _WaterBoundarySurface object or contain a
        _WaterBoundarySurface object inline, but neither both nor none. </xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="_WaterBoundarySurface"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="AbstractWaterBoundarySurfaceType" abstract="true">
    <xs:annotation>
      <xs:documentation>A WaterBoundarySurface is a thematic object which classifies surfaces bounding a water body.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="lod2Surface" type="gml:SurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod3Surface" type="gml:SurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod4Surface" type="gml:SurfacePropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfWaterBoundarySurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_WaterBoundarySurface" type="AbstractWaterBoundarySurfaceType" abstract="true"
    substitutionGroup="core:_CityObject"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfWaterBoundarySurface" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="WaterSurfaceType">
    <xs:annotation>
      <xs:documentation>Type describing the surface of a water body, which separates the water from the air. As subclass of
        _CityObject, a WaterSurface inherits all attributes and relations, in particular an id, names, external references, and
        generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractWaterBoundarySurfaceType">
        <xs:sequence>
          <xs:element name="waterLevel" type="gml:CodeType" minOccurs="0">
            <xs:annotation>
              <xs:documentation>Type for the specification of the level of a water surface. The optional attribute waterLevel of a
                WaterSurface can be used to describe the water level, for which the given 3D surface geometry was acquired. This
                is especially important, when the water body is influenced by the tide. The values of this type are defined in the
                XML file WaterLevelType.xml, according to the dictionary concept of GML3. </xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element ref="_GenericApplicationPropertyOfWaterSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="WaterSurface" type="WaterSurfaceType" substitutionGroup="_WaterBoundarySurface"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfWaterSurface" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="WaterGroundSurfaceType">
    <xs:annotation>
      <xs:documentation>Type describing the ground surface of a water body, i.e. the boundary to the digital terrain model. As
        subclass of _CityObject, a WaterGroundSurface inherits all attributes and relations, in particular an id, names, external
        references, and generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractWaterBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfWaterGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="WaterGroundSurface" type="WaterGroundSurfaceType" substitutionGroup="_WaterBoundarySurface"/>
  <!-- ========================================================================================== -->
```

```
  <xs:element name="_GenericApplicationPropertyOfWaterGroundSurface" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="WaterClosureSurfaceType">
    <xs:annotation>
      <xs:documentation>Type describing the closure surface between water bodys. As subclass of _CityObject, a WaterClosureSurface
        inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractWaterBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfWaterClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="WaterClosureSurface" type="WaterClosureSurfaceType" substitutionGroup="_WaterBoundarySurface"/>
  <!-- ========================================================================================== -->
  <xs:element name="_GenericApplicationPropertyOfWaterClosureSurface" type="xs:anyType" abstract="true"/>
</xs:schema>
```

A.14 TexturedSurface module [deprecated]

The CityGML *TexturedSurface* module is defined within the XML Schema definition file *texturedSurface.xsd*. The target namespace http://www.opengis.net/citygml/texturedsurface/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/texturedsurface/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/texturedsurface/2.0" elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="2.0.0">
  <xs:annotation>
    <xs:documentation>
  CityGML is an OGC Standard.
  Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/ .
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/2.0"
             schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xs:complexType name="TexturedSurfaceType">
    <xs:annotation>
      <xs:appinfo>deprecated</xs:appinfo>
      <xs:documentation>Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead. The
        concept of positioning textures on surfaces complies with the standard X3D. Because there has been no appropriate
        texturing concept in GML3, CityGML adds the class TexturedSurface to the geometry model of GML 3. A texture is specified
        as a raster image referenced by an URI, and can be an arbitrary resource, even in the internet. Textures are positioned by
        employing the concept of texture coordinates, i.e. each texture coordinate matches with exactly one 3D coordinate of the
        TexturedSurface. The use of texture coordinates allows an exact positioning and trimming of the texture on the surface
        geometry. Each surface may be assigned one or more appearances, each refering to one side of the surface.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:OrientableSurfaceType">
        <xs:sequence>
          <xs:element ref="appearance" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="TexturedSurface" type="TexturedSurfaceType" substitutionGroup="gml:OrientableSurface"/>
  <!-- ========================================================================================== -->
  <xs:element name="appearance" type="AppearancePropertyType"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="AppearancePropertyType">
    <xs:annotation>
      <xs:appinfo>deprecated</xs:appinfo>
      <xs:documentation>Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead. A
        property that has an _Appearance as its value domain, which can either be a Material (Color,...) or a Texture. The
        _Appearance Element can either be encapsulated in an element of this type or an XLink reference to a remote _Appearance
        element (where remote geometry elements are located in another document or elsewhere in the same document). Either the
        reference or the contained element must be given, but neither both nor none. The side of the surface the _Appearance
        refers to is given by the orientation attribute, which refers to the corresponding sign attribute of the orientable
        surface: + means the side with positive orientation, and - the side with negative orientation. </xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="_Appearance"/>
    </xs:sequence>
    <xs:attribute name="orientation" type="gml:SignType" default="+"/>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="AbstractAppearanceType" abstract="true">
    <xs:annotation>
      <xs:appinfo>deprecated</xs:appinfo>
      <xs:documentation>Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead. This
        abstract type is the parent type of MaterialType and SimpleTextureType. It is derived from gml:AbstractGMLType, thus it
        inherits the attribute gml:id and may be referenced by an appearanceProperty, although it is defined elsewhere in another
        appearanceProperty. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
```



```
      <xs:extension base="gml:AbstractGMLType"/>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="_Appearance" type="AbstractAppearanceType" abstract="true" substitutionGroup="gml:_GML"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="MaterialType">
    <xs:annotation>
      <xs:appinfo>deprecated</xs:appinfo>
      <xs:documentation>Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead. Adopted
        from X3D standard (http://www.web3d.org/x3d/) </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractAppearanceType">
        <xs:sequence>
          <xs:element name="shininess" type="core:doubleBetween0and1" minOccurs="0"/>
          <xs:element name="transparency" type="core:doubleBetween0and1" minOccurs="0"/>
          <xs:element name="ambientIntensity" type="core:doubleBetween0and1" minOccurs="0"/>
          <xs:element name="specularColor" type="Color" minOccurs="0"/>
          <xs:element name="diffuseColor" type="Color" minOccurs="0"/>
          <xs:element name="emissiveColor" type="Color" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="Material" type="MaterialType" substitutionGroup="_Appearance"/>
  <!-- ========================================================================================== -->
  <xs:complexType name="SimpleTextureType">
    <xs:annotation>
      <xs:appinfo>deprecated</xs:appinfo>
      <xs:documentation>Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead. Adopted
        from X3D standard (http://www.web3d.org/x3d/). ToDo: repeat </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractAppearanceType">
        <xs:sequence>
          <xs:element name="textureMap" type="xs:anyURI"/>
          <xs:element name="textureCoordinates" type="gml:doubleList"/>
          <xs:element name="textureType" type="TextureTypeType" minOccurs="0"/>
          <xs:element name="repeat" type="xs:boolean" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:element name="SimpleTexture" type="SimpleTextureType" substitutionGroup="_Appearance"/>
  <!-- ========================================================================================== -->
  <xs:simpleType name="TextureTypeType">
    <xs:annotation>
      <xs:appinfo>deprecated</xs:appinfo>
      <xs:documentation>Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead.
        Textures can be qualified by the attribute textureType. The textureType differentiates between textures, which are
        specific for a certain object and are only used for that object (specific), and prototypic textures being typical for that
        kind of object and are used many times for all objects of that kind (typical). A typical texture may be replaced by a
        specific, if available. Textures may also be classified as unknown. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="specific"/>
      <xs:enumeration value="typical"/>
      <xs:enumeration value="unknown"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- ========================================================================================== -->
  <xs:simpleType name="Color">
    <xs:annotation>
      <xs:appinfo>deprecated</xs:appinfo>
      <xs:documentation>Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead. List of
        three values (red, green, blue), separated by spaces. The values must be in the range between zero and one.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="core:doubleBetween0and1List">
      <xs:length value="3"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

A.15 Schematron rules on referential integrity

The CityGML Schema repository provides Schematron rules precisely describing referential integrity constraints on CityGML property elements denoting the relation between CityGML objects. If not stated otherwise, these property elements are generally restricted to either carry a reference to a remote object using the XLink concept of GML 3.1.1 (where remote objects are located in another document or elsewhere in the same document) or contain an object inline, but neither both nor none. Within this standard document, conformance requirements on referential integrity are stated within a separate clause at the end of each chapter covering a CityGML module (e.g., see clause 10.3.9 for the CityGML *Building* module).

Schematron is an auxiliary constraint language. The constraints apply to context nodes within a CityGML instance document (typically an XML element) which are addressed based on XPath criteria. The rules themselves are given as XPath expressions which are evaluated for each of these nodes in order to check their validity. A CityGML instance document is considered invalid if any Schematron rule is violated. Please note, that for version 2.0 of CityGML the provided Schematron rules only limit property elements to act in either by-reference or by-value mode, but neither both nor none. Neither further restrictions on property elements nor other conformance requirements are covered. In future versions of CityGML, further Schematron rules may be added.

In CityGML, the Schematron schema used to express referential integrity rules is based on the Schematron Assertion Language version 1.5 which is also employed by GML 3.1.1. The schema is shipped as a separate file within the CityGML schema package and is accessible by the name *referentialIntegrity.sch*. It may be used to check conformance requirements on referential integrity of CityGML property elements within a CityGML instance document according to the CityGML abstract test suite provided in Annex B. However, this requires an XML validator capable of automatically processing the Schematron rules provided by the schema. Otherwise, the Schematron code can be treated merely as a formal description of the required constraints.

The following excerpt of *referentialIntegrity.sch* illustrates the abstract base rule *hrefOrContent*. This rule states XPath expressions to ensure that target objects of property elements are either given by reference or by value. It is identically specified as the corresponding rule provided by the XSD schema file *gmlBase.xsd* of GML 3.1.1. Since the *hrefOrContent* rule is abstract, it does not apply to any specific context node within a CityGML instance document but is referenced by concrete rules within the schema to ensure consistency.

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://www.ascc.net/xml/schematron">
  <sch:title>Schematron constraints for CityGML 2.0</sch:title>
  <sch:ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>
  <sch:ns prefix="bldg" uri="http://www.opengis.net/citygml/building/2.0"/>
  …
  <sch:pattern name="Check either href or content not both">
    <sch:rule abstract="true" id="hrefOrContent">
      <sch:report test="@xlink:href and (*|text())">Property element may not carry both a reference to an object and contain an
        object.</sch:report>
      <sch:assert test="@xlink:href | (*|text())">Property element must either carry a reference to an object or contain an
        object.</sch:assert>
    </sch:rule>
  </sch:pattern>
  …
</sch:schema>
```

For each CityGML property element being subject to referential integrity requirements, *referentialIntegrity.sch* contains further concrete rules based on the abstract rule *hrefOrContent*. For example, the following schema snippet applies the *hrefOrContent* rule to the property element *bldg:consistsOfBuildingPart* defined within the CityGML *Building* module (cf. chapter 10.3). The *bldg:consistsOfBuildingPart* element denotes the relation of a *bldg:_AbstractBuilding* to its building part objects which may only be given inline or by reference.

```
  …
  <sch:pattern name="hrefOrContent check on bldg:consistsOfBuildingPart">
    <sch:rule context="bldg:consistsOfBuildingPart">
      <sch:extends rule="hrefOrContent"/>
    </sch:rule>
  </sch:pattern>
  …
```


Annex B
(normative)

Abstract test suite for CityGML instance documents

B.1 Test cases for mandatory conformance requirements

B.1.1 Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of each CityGML module that is part of the CityGML profile employed by the instance document. This may be any combination of CityGML extension modules in conjunction with the CityGML core module.
b) Test method	Validate the CityGML XML instance document against the XML Schema definitions of all employed CityGML modules. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the respective XML Schema specification of the employed CityGML modules.
c) Reference	Annex A.
d) Test type	Basic Test.

B.1.2 Valid CityGML profile

CityGML profile definition embedded inline the CityGML instance document

a) Test purpose	Verify that a profile employed by a CityGML instance document is a valid CityGML profile in accordance with the rules and guidelines stated in chapter 7.2. For CityGML profile definitions embedded inline the CityGML instance document (referenced as first approach in chapter 7.2), verify that the CityGML instance document denotes all schema definitions and corresponding XML namespaces of CityGML modules that are used to represent the data within the instance document and, thus, are part of the employed CityGML profile.
b) Test method	Inspect the instance document and check that it satifies the rules for employing CityGML profiles described in chapter 7.2 (first approach).
c) Reference	Annex A, chapter 7.2 (first approach).
d) Test type	Basic Test.

CityGML profile definition provided by a separate XML Schema definition file

a) Test purpose	Verify that a profile employed by a CityGML instance document is a valid CityGML profile in accordance with the rules and guidelines stated in chapter 7.2. For CityGML profile definitions provided by a separate XML Schema definition file (referenced as second approach in chapter 7.2), verify that the profile's XML Schema definition file itself is valid and imports all schema definitions of CityGML modules that are used to represent the data within the instance document and, thus, are part of the employed CityGML profile. The target namespace of the profile's XML Schema definition must differ from the namespaces of the imported CityGML modules and must be given as previously unused and globally unique URI. The profile's XML Schema definition must not contain any further content.
b) Test method	Validate the XML Schema definition of the CityGML profile. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the respective XML Schema specification of the CityGML profile. Inspect

	the instance document and check that it satifies the rules for employing CityGML profiles described in chapter 7.2 (second approach).
c) Reference	Annex A, chapter 7.2 (second approach).
d) Test type	Basic Test.

B.1.3 Conformance classes related to CityGML modules

a) Test purpose	Verify the validity of the CityGML instance document against the conformance classes of each CityGML module that is part of the CityGML profile employed by the instance document. This may be any combination of CityGML extension modules in conjunction with the CityGML core module.
b) Test method	Follow the test cases provided by the conformace classes for each CityGML module in annex B.2.
c) Reference	Annex B.2.
d) Test type	Basic Test.

B.1.4 Spatial geometry objects

a) Test purpose	Verify that all spatial geometry objects within a CityGML instance document adhere to the XML Schema definition of the Geography Markup Language version 3.1.1 and to the CityGML spatial model.
b) Test method	Inspect the instance document and check that spatial geometry objects are valid with respect to the XML Schema definition of GML version 3.1.1 and satisfy the rules of to the CityGML spatial model described in chapter 8.
c) Reference	OGC Document No. 03-105r1, Annex A, chapter 8.
d) Test type	Capability Test.

B.1.5 Spatial topology relations

a) Test purpose	Verify that all spatial topology relations between spatial geometry objects are expressed using the XML concept of *XLinks* provided by GML version 3.1.1.
b) Test method	Inspect the instance document and check that spatial topology relations between spatial geometry objects are valid with respect to the *XLinks* concept introduced by GML version 3.1.1 and satisfy the rules of to the CityGML spatial model described in chapter 8.
c) Reference	OGC Document No. 03-105r1, Annex A, chapter 8.
d) Test type	Capability Test.

B.1.6 Address objects

a) Test purpose	Verify that all thematic objects representing address information within a CityGML instance document adhere to the XML Schema definition of the Extensible Address Language (xAL) issued by OASIS and to the rules for representing address information in CityGML.
b) Test method	Inspect the instance document and check that thematic objects representing address information are valid with respect to the XML Schema definition of the OASIS Extensible Address Language and satisfy the rules for representing address information within CityGML described in chapter 10.1.4.
c) Reference	OASIS 2003, Annex A, chapter 10.1.4.
d) Test type	Capability Test.

B.2 Conformance classes related to CityGML modules

B.2.1 CityGML Core module

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *CityGML Core* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents.
b) Test method	Inspect the instance document and check that it satifies the rules of the *CityGML Core* module described in chapter 10.1 and 8.2, and especially the conformance requirements stated in chapter 10.1.6 and 8.3.3. Conformance requirements on referential integrity of CityGML property elements defined within the *CityGML Core* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.
c) Reference	Chapter 10.1, 10.1.6, 8.2, 8.3.3, annex A.15.
d) Test type	Basic Test.

Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *CityGML Core* module. This test case is mandatory for all CityGML instance documents.
b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *CityGML Core* module in annex A.1. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *CityGML Core* module.
c) Reference	Annex A.1.
d) Test type	Basic Test.

B.2.2 Appearance module

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *Appearance* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *Appearance* module. Conformance requirements on referential integrity of CityGML property elements defined within the *Appearance* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.
b) Test method	Inspect the instance document and check that it satifies the rules of the *Appearance* module described in chapter 9, and especially the conformance requirements stated in chapter 9.7.
c) Reference	Chapter 9, 9.7, annex A.15.
d) Test type	Capability Test.

Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *Appearance* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *Appearance* module.

b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *Appearance* module in annex A.2. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *Appearance* module.
c) Reference	Annex A.2.
d) Test type	Capability Test.

B.2.3 Bridge module

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *Bridge* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *Bridge* module. Conformance requirements on referential integrity of CityGML property elements defined within the *Bridge* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.
b) Test method	Inspect the instance document and check that it satifies the rules of the *Bridge* module described in chapter 10.5, and especially the conformance requirements stated in chapter 10.5.8.
c) Reference	Chapter 10.5, 10.5.8, annex A.15.
d) Test type	Capability Test.

Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *Bridge* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *Bridge* module.
b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *Bridge* module in annex A.3. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *Bridge* module.
c) Reference	Annex A.3.
d) Test type	Capability Test.

B.2.4 Building module

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *Building* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *Building* module. Conformance requirements on referential integrity of CityGML property elements defined within the *Building* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.
b) Test method	Inspect the instance document and check that it satifies the rules of the *Building* module described in chapter 10.3, and especially the conformance requirements stated in chapter 10.3.9.

c) Reference	Chapter 10.3, 10.3.9, annex A.15.
d) Test type	Capability Test.

Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *Building* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *Building* module.
b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *Building* module in annex A.3. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *Building* module.
c) Reference	Annex A.4.
d) Test type	Capability Test.

B.2.5 CityFurniture module

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *CityFurniture* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *CityFurniture* module. Conformance requirements on referential integrity of CityGML property elements defined within the *CityFurniture* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.
b) Test method	Inspect the instance document and check that it satifies the rules of the *CityFurniture* module described in chapter 10.9, and especially the conformance requirements stated in chapter 10.9.4.
c) Reference	Chapter 10.9, 10.9.4, annex A.15.
d) Test type	Capability Test.

Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *CityFurniture* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *CityFurniture* module.
b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *CityFurniture* module in annex A.5. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *CityFurniture* module.
c) Reference	Annex A.5.
d) Test type	Capability Test.

B.2.6 CityObjectGroup module

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *CityObjectGroup* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This

	test case is mandatory for all CityGML instance documents employing elements defined within the *CityObjectGroup* module. Conformance requirements on referential integrity of CityGML property elements defined within the *CityObjectGroup* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.
b) Test method	Inspect the instance document and check that it satifies the rules of the *CityObjectGroup* module described in chapter 10.11, and especially the conformance requirements stated in chapter 10.11.2.
c) Reference	Chapter 10.11, 10.11.2, annex A.15.
d) Test type	Capability Test.

Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *CityObjectGroup* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *CityObjectGroup* module.
b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *CityObjectGroup* module in annex A.6. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *CityObjectGroup* module.
c) Reference	Annex A.6.
d) Test type	Capability Test.

B.2.7 Generics module

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *Generics* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *Generics* module. Conformance requirements on referential integrity of CityGML property elements defined within the *Generics* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.
b) Test method	Inspect the instance document and check that it satifies the rules of the *Generics* module described in chapter 10.12, and especially the conformance requirements stated in chapter 10.12.2.
c) Reference	Chapter 10.12, 10.12.2, annex A.15.
d) Test type	Capability Test.

Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *Generics* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *Generics* module.
b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *Generics* module in annex A.7. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *Generics* module.
c) Reference	Annex A.7.

d) Test type	Capability Test.

B.2.8 LandUse module

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *LandUse* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *LandUse* module. Conformance requirements on referential integrity of CityGML property elements defined within the *LandUse* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.
b) Test method	Inspect the instance document and check that it satifies the rules of the *LandUse* module described in chapter 10.10, and especially the conformance requirements stated in chapter 10.10.3.
c) Reference	Chapter 10.10, 10.10.3, annex A.15.
d) Test type	Capability Test.

Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *LandUse* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *LandUse* module.
b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *LandUse* module in annex A.8. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *LandUse* module.
c) Reference	Annex A.8.
d) Test type	Capability Test.

B.2.9 Relief module

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *Relief* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *Relief* module. Conformance requirements on referential integrity of CityGML property elements defined within the *Relief* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.
b) Test method	Inspect the instance document and check that it satifies the rules of the *Relief* module described in chapter 10.2, and especially the conformance requirements stated in chapter 10.2.6.
c) Reference	Chapter 10.2, 10.2.6, annex A.15.
d) Test type	Capability Test.

Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *Relief* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *Relief* module.
b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *Relief* module in annex A.9. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *Relief* module.
c) Reference	Annex A.9.
d) Test type	Capability Test.

B.2.10 Transportation module

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *Transportation* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *Transportation* module. Conformance requirements on referential integrity of CityGML property elements defined within the *Transportation* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.
b) Test method	Inspect the instance document and check that it satifies the rules of the *Transportation* module described in chapter 10.7, and especially the conformance requirements stated in chapter 10.7.5.
c) Reference	Chapter 10.7, 10.7.5, annex A.15.
d) Test type	Capability Test.

Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *Transportation* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *Transportation* module.
b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *Transportation* module in annex A.10. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *Transportation* module.
c) Reference	Annex A.10.
d) Test type	Capability Test.

B.2.11 Tunnel module

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *Tunnel* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *Tunnel* module. Conformance requirements on referential integrity of CityGML property elements defined within the *Tunnel* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.

b) Test method	Inspect the instance document and check that it satifies the rules of the *Tunnel* module described in chapter 10.4, and especially the conformance requirements stated in chapter 10.4.8.
c) Reference	Chapter 10.4, 10.4.8, annex A.15.
d) Test type	Capability Test.

Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *Tunnel* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *Tunnel* module.
b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *Tunnel* module in annex A.3. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *Tunnel* module.
c) Reference	Annex A.11.
d) Test type	Capability Test.

B.2.12 Vegetation module

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *Vegetation* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *Vegetation* module. Conformance requirements on referential integrity of CityGML property elements defined within the *Vegetation* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.
b) Test method	Inspect the instance document and check that it satifies the rules of the *Vegetation* module described in chapter 10.8, and especially the conformance requirements stated in chapter 10.8.6.
c) Reference	Chapter 10.8, 10.8.6, annex A.15.
d) Test type	Capability Test.

Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *Vegetation* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *Vegetation* module.
b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *Vegetation* module in annex A.12. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *Vegetation* module.
c) Reference	Annex A.12.
d) Test type	Capability Test.

B.2.13 WaterBody module

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *WaterBody* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *WaterBody* module. Conformance requirements on referential integrity of CityGML property elements defined within the *WaterBody* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.
b) Test method	Inspect the instance document and check that it satifies the rules of the *WaterBody* module described in chapter 10.6, and especially the conformance requirements stated in chapter 10.6.4.
c) Reference	Chapter 10.6, 10.6.4, annex A.15.
d) Test type	Capability Test.

Valid CityGML instance document

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *WaterBody* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *WaterBody* module.
b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *WaterBody* module in annex A.13. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *WaterBody* module.
c) Reference	Annex A.13.
d) Test type	Capability Test.

B.2.14 TexturedSurface module [deprecated]

Mandatory conformance requirements

a) Test purpose	Verify that the CityGML instance document follows the *TexturedSurface* module's rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *TexturedSurface* module. Conformance requirements on referential integrity of CityGML property elements defined within the *TexturedSurface* module may be additionally validated using the constraints provided by the Schematron schema *referentialIntegrity.sch* in accordance with the rules and guidelines stated in annex A.15.
b) Test method	Inspect the instance document and check that it satifies the rules of the *TexturedSurface* module described in chapter 9.8, and especially the conformance requirements stated in chapter 9.8.2.
c) Reference	Chapter 9.8, 9.8.2, annex A.15.
d) Test type	Capability Test.

Valid CityGML instance documents

a) Test purpose	Verify the validity of the CityGML instance document against the XML Schema definition of the *TexturedSurface* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *TexturedSurface* module.

b) Test method	Validate the CityGML XML instance document against the XML Schema definition of the *TexturedSurface* module in annex A.14. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *TexturedSurface* module.
c) Reference	Annex A.14.
d) Test type	Capability Test.

Annex C
(informative)

Code lists proposed by the SIG 3D

This annex exemplifies the specification of code lists for enumerative attributes of type *gml:CodeType* and provides proposals for selected attributes. Please note that this annex is non-normative and the presented code lists are neither mandatory nor complete. The governance of code lists is explicitly decoupled from the governance of the CityGML schema and specification (cf. chapter 10.14). Thus, code lists can be freely defined outside the CityGML specification by any organization or information community according to their information needs, and there shall be one authority per code list who is in charge of maintance and updating the contents.

The code lists presented in this annex are managed and maintained by the SIG 3D and are accessible at http://www.sig3d.de/codelists/standard/. They completely comprise the contents of the non-normative code lists given in the previous CityGML 1.0 specification document (cf. Annex C of OGC Doc. No. 08-007r1). In order to maintain backwards compatibility, none of the previous entries has been modified or changed in meaning. Additional code lists have been added for some of the newly introduced enumerative attributes in CityGML 2.0, for example in the context of the *Bridge* and *Tunnel* module. The SIG 3D code lists will be complemented subsequently. The previous version 1.0 code lists are still accessible from the OGC repository at http://schemas.opengis.net/citygml/codelists/.

The SIG 3D code lists are implemented as *simple dictionaries* following the *GML 3.1.1 Simple Dictionary Profile* and can be directly referenced in CityGML instance documents. For this purpose, the *codeSpace* attribute declared by *gml:CodeType* has to point to the code list of the corresponding enumerative feature attribute. The following naming pattern is applied by the SIG 3D in order to derive a globally unique URL in the *http* schema for each code list:

http://www.sig3d.org/codelists/standard/
<moduleName>/<version>/<FeatureTypeName>_<propertyName>.xml

The placeholder *<moduleName>* has to be replaced with the name of the CityGML module in which the feature and the attribute are defined. The module name has to be given in small letters (e.g., *building* or *cityfurniture*). The *<version>* field defines the CityGML version for which the code list is specified. Finally, *<FeatureTypeName>* denotes the name of the feature type for which the attribute is modeled and *<propertyName>* is to be replaced with the attribute name itself. For both placeholders, the XML element name given in the CityGML schema document where the feature is specified has to be chosen.

For example, the *codeSpace* value identifying the SIG 3D code list for the enumerative attribute *roofType* of the feature *_AbstractBuilding* (*Building* module, cf. chapter 10.3) is:

http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml

The following excerpt demonstrates how to reference the SIG 3D code list for *roofType* from within a CityGML instance document. The attribute value *1000* denotes a *flat roof* according to this code list (cf. subclause C.1).

```
<bldg:Building>
  <bldg:roofType
      codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml">1000</bldg:roofType>
  …
</bldg:Building>
```

The SIG 3D code lists can be downloaded from these URLs in order to allow applications to automatically validate the coded attribute values.

The entire code list for building roof types is given as example below, while the others are depicted in the following subclauses in tabular form for space reasons. The subclauses are arranged according to thematic modules.

Code list for roof types *(http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml)*

```
<gml:Dictionary xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/gml
  http://schemas.opengis.net/gml/3.1.1/profiles/SimpleDictionary/1.0.0/gmlSimpleDictionaryProfile.xsd"
  gml:id="_AbstractBuilding_roofType">
  <gml:name>_AbstractBuilding_roofType</gml:name>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id1">
      <gml:description>flat roof</gml:description>
      <gml:name>1000</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id2">
      <gml:description>monopitch roof</gml:description>
      <gml:name>1010</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id3">
      <gml:description>skip pent roof</gml:description>
      <gml:name>1020</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id4">
      <gml:description>gabled roof</gml:description>
      <gml:name>1030</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id5">
      <gml:description>hipped roof</gml:description>
      <gml:name>1040</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id6">
      <gml:description>half-hipped roof</gml:description>
      <gml:name>1050</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id7">
      <gml:description>mansard roof</gml:description>
      <gml:name>1060</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id8">
      <gml:description>pavilion roof</gml:description>
      <gml:name>1070</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id9">
      <gml:description>cone roof</gml:description>
      <gml:name>1080</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id10">
      <gml:description>copula roof</gml:description>
      <gml:name>1090</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id11">
      <gml:description>shed roof</gml:description>
      <gml:name>1100</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id12">
```

```
        <gml:description>arch roof</gml:description>
        <gml:name>1110</gml:name>
      </gml:Definition>
    </gml:dictionaryEntry>
    <gml:dictionaryEntry>
      <gml:Definition gml:id="id13">
        <gml:description>pyramidal broach roof</gml:description>
        <gml:name>1120</gml:name>
      </gml:Definition>
    </gml:dictionaryEntry>
    <gml:dictionaryEntry>
      <gml:Definition gml:id="id14">
        <gml:description>combination of roof forms</gml:description>
        <gml:name>1130</gml:name>
      </gml:Definition>
    </gml:dictionaryEntry>
</gml:Dictionary>
```

C.1 Building module

Code list of the *_AbstractBuilding* attribute *class*			
http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_class.xml			
1000	habitation	1100	schools, education, research
1010	sanitation	1110	maintainence and waste management
1020	administration	1120	healthcare
1030	business, trade	1130	communicating
1040	catering	1140	security
1050	recreation	1150	storage
1060	sport	1160	industry
1070	culture	1170	traffic
1080	church institution	1180	function
1090	agriculture, forestry		

Code list of the *_AbstractBuilding* attributes *function* and *usage*			
http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_function.xml http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_usage.xml			
1000	residential building	1840	rubbish bunker
1010	tenement	1850	building for rubbish incineration
1020	hostel	1860	building for rubbish disposal
1030	residential- and administration building	1870	building for agrarian and forestry
1040	residential- and office building	1880	barn
1050	residential- and business building	1890	stall
1060	residential- and plant building	1900	equestrian hall
1070	agrarian- and forestry building	1910	alpine cabin
1080	residential- and commercial building	1920	hunting lodge
1090	forester's lodge	1930	arboretum
1100	holiday house	1940	glass house
1110	summer house	1950	moveable glass house
1120	office building	1960	public building
1130	credit institution	1970	administration building
1140	insurance	1980	parliament
1150	business building	1990	guildhall
1160	department store	2000	post office
1170	shopping centre	2010	customs office
1180	kiosk	2020	court
1190	pharmacy	2030	embassy or consulate
1200	pavilion	2040	district administration
1210	hotel	2050	district government
1220	youth hostel	2060	tax office
1230	campsite building	2070	building for education and research
1240	restaurant	2080	comprehensive school
1250	cantine	2090	vocational school
1260	recreational site	2100	college or university
1270	function room	2110	research establishment
1280	cinema	2120	building for cultural purposes
1290	bowling alley	2130	castle
1300	casino	2140	theatre or opera
1310	industrial building	2150	concert building
1320	factory	2160	museum

1330	workshop	2170	broadcasting building
1340	petrol / gas station	2180	activity building
1350	washing plant	2190	library
1360	cold store	2200	fort
1370	depot	2210	religious building
1380	building for research purposes	2220	church
1390	quarry	2230	synagogue
1400	salt works	2240	chapel
1410	miscellaneous industrial building	2250	community center
1420	mill	2260	place of worship
1430	windmill	2270	mosque
1440	water mill	2280	temple
1450	bucket elevator	2290	convent
1460	weather station	2300	building for health care
1470	traffic assets office	2310	hospital
1480	street maintenance	2320	healing centre or care home
1490	waiting hall	2330	health centre or outpatients clinic
1500	signal control box	2340	building for social purposes
1510	engine shed	2350	youth centre
1520	signal box or stop signal	2360	seniors centre
1530	plant building for air traffic	2370	homeless shelter
1540	hangar	2380	kindergarten or nursery
1550	plant building for shipping	2390	asylum seekers home
1560	shipyard	2400	police station
1570	dock	2410	fire station
1580	plant building for canal lock	2420	barracks
1590	boathouse	2430	bunker
1600	plant building for cablecar	2440	penitentiary or prison
1610	multi-storey car park	2450	cemetery building
1620	parking level	2460	funeral parlor
1630	garage	2470	crematorium
1640	vehicle hall	2480	train station
1650	underground garage	2490	airport building
1660	building for supply	2500	building for underground station
1670	waterworks	2510	building for tramway
1680	pump station	2520	building for bus station
1690	water basin	2530	shipping terminal
1700	electric power station	2540	building for recuperation purposes
1710	transformer station	2550	building for sport purposes
1720	converter	2560	sports hall
1730	reactor	2570	building for sports field
1740	turbine house	2580	swimming baths
1750	boiler house	2590	indoor swimming pool
1760	building for telecommunications	2600	sanatorium
1770	gas works	2610	zoo building
1780	heat plant	2620	green house
1790	pumping station	2630	botanical show house
1800	building for disposal	2640	bothy
1810	building for effluent disposal	2650	tourist information centre
1820	building for filter plant	2700	others
1830	toilet		

Code list of the *BuildingFurniture* attribute *class*			
http://www.sig3d.org/codelists/standard/building/2.0/BuildingFurniture_class.xml			
1000	habitation	1100	schools, education, research
1010	sanitation	1110	maintenance, waste management
1020	administration	1120	healthcare
1030	business, trade	1130	communicating
1040	catering	1140	security
1050	recreation	1150	storage
1060	sport	1160	industry
1070	culture	1170	traffic
1080	church institution	1180	function
1090	agriculture, forestry		

Code list of the *BuildingFurniture* attributes *function* and *usage*			
http://www.sig3d.org/codelists/standard/building/2.0/BuildingFurniture_function.xml http://www.sig3d.org/codelists/standard/building/2.0/BuildingFurniture_usage.xml			
1000	cupboard	2010	sink, hand-basin
1010	wardrobe	2020	water tap
1020	cabinet	2030	toilet bowl
1030	sideboard	2040	bathtub
1040	locker	2050	shower
1050	tool cabinet	2060	bidet
1100	shelf	2100	animal park
1110	rack	2110	aquarium
1120	coat stand	2120	cage
1200	table	2130	birdcage
1210	dining table	2200	religious equipment
1220	coffee table	2300	shop fittings
1230	desk	2310	sales counter
1240	bedside cabinet	2320	glass cabinet
1250	baby changing table	2330	changing cubicle
1260	bar	2340	refrigerated counter
1270	pool table	2350	cash desk or till or counter
1280	snooker table	2360	box-office
1290	roulette table	2400	machines
1270	work bench	2410	ticket machine
1300	chair	2420	cigarette machine
1310	bench	2430	cash machine or ATM
1320	office chair	2440	vending machine
1330	sofa	2450	gambling machine
1340	rocking chair	2500	technical furniture
1350	bar stool	2510	heating installation
1360	armchair	2520	tank
1400	bed	2521	oil tank
1410	crib	2522	water tank
1420	bunk bed	2523	gas tank
1430	cradle	2524	fuel tank
1440	cot	2525	milk tank
1450	stretcher	2526	steel tank
1500	lighting	2530	fire protection appliance
1510	standard lamp	2531	fire extinguishing system
1520	ceiling light	2532	fire alarm

1530	spotlight	2533	fire extinguisher
1600	electric appliances	2540	switch board
1610	television set	2550	lifting platform
1620	video recorder	2560	compressed air system
1630	stereo unit	2570	loud-speaker
1700	kitchen appliances	2580	microphone
1710	cooker	2600	sports equipment
1720	oven	2610	goal posts
1730	refrigerator	2620	basketball basket
1740	coffee machine	2630	volleyball net
1750	toaster	2640	gymnastic apparatus
1760	kettle	2650	diving platform
1770	microwave	2660	swimming pool
1780	dish washer	2700	sales promotion furniture
1800	laundry equipment	2710	display panel
1810	washing machine	2720	billboard
1820	ironing machine	2730	display cabinet
1830	rotary iron (mangle)	2800	functional furniture
1840	laundry tumble drier	2805	ashtray
1850	spin drier	2810	lectern
1900	technical office equipment	2815	stage
1910	copy machine	2820	blackboard
1920	scanner	2825	screen
1930	plotter	2830	mapstand
1940	printer	2835	rubbish bin
1950	screen	2840	sauna
1960	computer	2845	carpet
1970	overhead projector	2850	wall clock
1980	video projector	2855	curtain
2000	sanitation equipment	2860	mirror

Code list of the *BuildingInstallation* attribute *class*			
http://www.sig3d.org/codelists/standard/building/2.0/BuildingInstallation_class.xml			
1000	outer characteristics	1040	communicating
1010	inner characteristics	1050	security
1020	waste management	1060	others
1030	maintenance		

Code list of the *BuildingInstallation* attributes *function* and *usage*			
http://www.sig3d.org/codelists/standard/building/2.0/BuildingInstallation_function.xml http://www.sig3d.org/codelists/standard/building/2.0/BuildingInstallation_usage.xml			
1000	balcony	1040	tower (part of a building)
1010	winter garden	1050	column
1020	arcade	1060	stairs
1030	chimney (part of a building)	1070	others

Code list of the *IntBuildingInstallation* attribute *class*			
http://www.sig3d.org/codelists/standard/building/2.0/IntBuildingInstallation_class.xml			
1000	Heating, Ventilation, Climate	6000	Statics
2000	Safety	7000	Entertainmant
3000	Illumination	8000	Miscellaneous

4000	Communication	9999	Unknown
5000	Supply and Disposal		

Code list of the *IntBuildingInstallation* attributes *function* and *usage*			
http://www.sig3d.org/codelists/standard/building/2.0/IntBuildingInstallation_function.xml http://www.sig3d.org/codelists/standard/building/2.0/IntBuildingInstallation_usage.xml			
1010	Radiator	3020	Light switch
1020	Oven	5030	Power point
1030	Fireside	5020	Cable
1040	Ventilator	7010	Rafter
1050	Air Conditioning	7020	Column
5010	Pipe	8010	Railing
3010	Lamp	8020	Stair

Code list of the *_AbstractBuilding* attribute *roofType*			
http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml			
1000	flat roof	1070	pavilion roof
1010	monopitch roof	1080	cone roof
1020	dual pent roof	1090	copula roof
1030	gabled roof	1100	sawtooth roof
1040	hipped roof	1110	arch roof
1050	half-hipped roof	1120	pyramidal broach roof
1060	mansard roof	1130	combination of roof forms

Code list of the *Room* attribute *class*			
http://www.sig3d.org/codelists/standard/building/2.0/Room_class.xml			
1000	habitation	1080	accommodation, waste management
1010	administration	1090	healthcare
1020	business, trade	1100	communicating
1030	catering	1110	security
1040	recreation	1120	store
1050	church institution	1130	industry
1060	agriculture, forestry	1140	traffic
1070	schools, education, research	1150	function

Code list of the *Room* attributes *function* and *usage*			
http://www.sig3d.org/codelists/standard/building/2.0/Room_function.xml http://www.sig3d.org/codelists/standard/building/2.0/Room_usage.xml			
1000	living room	2170	showers
1010	bedroom	2200	tribune
1020	kitchen	2210	seating / standing capacity
1030	hall	2220	cash point
1040	bath, washroom	2230	vivarium
1050	toilet	2240	enclosure
1060	stairs	2250	aquarium
1070	home office	2260	terrarium
1080	utility room	2270	aviary
1090	dining room	2280	menagerie
1100	common room	2290	stables
1110	party room	2300	greenhouse
1120	nursery	2310	food silo

1130	store room	2320	hayloft
1140	canteen, common kitchen	2330	motor pool
1150	storeroom	2340	barn
1160	balcony, gallery	2350	riding hall
1170	terrace	2360	horse box
1180	drying room	2370	hunting lodge
1190	heatingroom	2400	waste container
1200	fuel depot	2410	motor pool
1210	hobby room	2420	washing-bay
1220	stable, hovel	2430	installations room
1300	cash office	2440	monitoring room
1310	ticket office	2450	heating system
1320	conference room	2460	public utility use
1330	reception	2470	pump room
1340	sales room	2480	effluent treatment
1350	store room	2490	treatment installation
1360	delivery	2500	recycling installation
1370	lounge, common room	2600	chancel
1380	escalator	2610	sacristy
1390	guest toilet	2620	bell tower
1400	strong room	2630	baptism room
1500	office	2640	confessional
1510	entrance hall	2650	benches
1520	elevator	2660	pulpit
1530	canteen	2670	lobby
1540	tea kitchen / Coffee kitchen	2680	parish
1550	archive	2690	chapel
1560	citizen office	2700	police station
1570	conference hall	2710	headquarters
1580	copier room / blueprint room	2720	prison cell
1590	information	2730	motor pool hall
1600	computer room	2740	fire brigade, emergency vehicle
1610	printer / plotter room	2750	relaxation room
1700	reception	2760	tool / pipe store
1710	guest room	2770	emergency call center
1720	bar	2780	arms depot
1730	breakfast room	2790	ammunition dump
1740	dining room	2800	vehicle hall
1750	celebration room	2810	panic room
1760	pub	2900	satellite receiver
1770	beer garden	2910	communication room
1780	restaurant	3000	industrial building
1790	cool store	3010	production building
1800	bowling alley, shoot alley	3020	factory building
1810	lounge	3030	workshop
1820	canteen kitchen	3040	storage depot
1900	stage	3050	cold storage
1910	auditorium	3060	store
1920	VIP box	3100	station concourse
1930	projection room	3110	track
1940	dressing room	3120	ticket office
1950	cabin	3130	waiting hall
1960	showroom	3140	engine shed

1970	equipment or props	3150	signal box
1980	make-up room	3160	departure terminal
1990	recording studio	3170	check-out counter
2000	sound studio	3180	check-in counter
2010	music archive	3190	check
2020	administration	3200	baggage carousel
2030	ticket office	3210	security check
2040	library	3300	classroom
2050	media room	3310	staff room
2060	dressing room	3320	break or recess hall
2070	sport room	3330	laboratory
2080	equipment room	3340	utility room
2090	platform	3350	media room
2100	swimming-pool	3360	science laboratory
2110	slide	3370	sports hall
2120	relaxation room	3380	school library
2130	sauna	3390	office
2140	fitness room	3400	lecture theatre
2150	solarium	3410	refectory
2160	catering	3420	function room

C.2 Tunnel module

Code list of the *_AbstractTunnel* attribute *class*			
http://www.sig3d.org/codelists/standard/tunnel/2.0/_AbstractTunnel_class.xml			
1000	traffic	1030	others
1010	supply		
1020	historical		

Code list of the *_AbstractTunnel* attributes *function* and *usage*			
http://www.sig3d.org/codelists/standard/tunnel/2.0/_AbstractTunnel_function.xml http://www.sig3d.org/codelists/standard/tunnel/2.0/_AbstractTunnel_usage.xml			
1000	railway tunnel	1020	canal tunnel
1010	roadway tunnel	1030	pedestrian tunnel

C.3 Bridge module

Code list of the *_AbstractBridge* attribute *class*			
http://www.sig3d.org/codelists/standard/bridge/2.0/_AbstractBridge_class.xml			
1000	arced bridge	1040	truss bridge
1010	cable-stayed bridge	1050	pontoon bridge
1020	deck bridge	1060	suspension bridge
1030	cable-stayed overpass		

Code list of the *_AbstractBridge* attributes *function* and *usage*			
http://www.sig3d.org/codelists/standard/bridge/2.0/_AbstractBridge_function.xml http://www.sig3d.org/codelists/standard/bridge/2.0/_AbstractBridge_usage.xml			
1000	railway bridge	1040	canal bridge
1010	roadway bridge	1050	aqueduct
1030	cable link	1060	foot bridge

C.4 CityFurniture module

Code list of the *CityFurniture* attribute *class*			
http://www.sig3d.org/codelists/standard/cityfurniture/2.0/CityFurniture_class.xml			
1000	traffic	1020	security
1010	communication	1030	others

Code list of the *CityFurniture* attributes *function* and *usage*			
http://www.sig3d.org/codelists/standard/cityfurniture/2.0/CityFurniture_function.xml http://www.sig3d.org/codelists/standard/cityfurniture/2.0/CityFurniture_usage.xml			
1000	communication fixture	1270	pole
1010	telephone box	1280	radio mast
1020	postbox	1290	aerial
1030	emergency call fixture	1300	radio telescope
1040	fire detector	1310	chimney
1050	police call post	1320	marker
1060	switching unit	1330	hydrant
1070	road sign	1340	upper corridor fire-hydrant
1080	traffic light	1350	lower floor panel fire-hydrant
1090	free-standing sign	1360	slidegate valve cap
1100	free-standing warning sign	1370	entrance shaft
1110	bus stop	1380	converter
1120	milestone	1390	stair
1130	rail level crossing	1400	outside staircase
1140	gate	1410	escalator
1150	streetlamp, latern or candelabra	1420	ramp
1160	column	1430	patio
1170	lamp post	1440	fence
1180	flagpole	1450	memorial/monument
1190	street sink box	1470	wayside shrine
1200	rubbish bin	1480	crossroads
1210	clock	1490	cross on the summit of a mountain
1220	directional spot light	1500	fountain
1230	floodlight mast	1510	block mark
1240	windmill	1520	boundary post
1250	solar cell	1530	bench
1260	water wheel	1540	others

C.5 LandUse module

Code list of the *LandUse* attribute *class*			
http://www.sig3d.org/codelists/standard/landuse/2.0/LandUse_class.xml			
1000	Settlement Area	3000	Vegetation
1100	Undeveloped Area	4000	Water
2000	Traffic		

Code list of the *LandUse* attributes *function* and *usage*			
http://www.sig3d.org/codelists/standard/landuse/2.0/LandUse_function.xml http://www.sig3d.org/codelists/standard/landuse/2.0/LandUse_usage.xml			
1010	Residential	2050	Track
1020	Industry and Business	2060	Square
1030	Mixed use	3010	Grassland
1040	Special Function Area	3020	Agriculture
1050	Monument	3030	Forest
1060	Dump	3040	Grove
1070	Mining	3050	Heath
1110	Park	3060	Moor
1120	Cemetary	3070	Marsh
1130	Sports, leisure and recreation	3080	Untilled land
1140	Open pit, quarry	4010	River
2010	Road	4020	Standing Waterbody
2020	Railway	4030	Harbour
2030	Airfield	4040	Sea
2040	Shipping		

C.6 Mime types

Codelist of the *ImplicitGeometry* (*Core* module) / *_Texture* (*Appearance* module) attribute *mimeType*			
http://www.sig3d.org/codelists/standard/core/2.0/ImplicitGeometry_mimeType.xml http://www.sig3d.org/codelists/standard/appearance/2.0/_Texture_mimeType.xml			
The MIME types given in this table are defined by the Internet Assigned Numbers Authority (IANA), see http://www.iana.org/. Generally, the MIME format is standardized by the Internet Engineering Task Force (IETF), see http://www.ietf.org/. Unlike the other code lists the MIME types are not represented by numbers, but instead use their given identifier. This code list is not exhaustive. It contains a selection of frequently used MIME types.			
model/vrml	VRML97	model/x3d+xml	X3D
application/x-3ds	3ds max	model/x3d+binary	X3D
application/dxf	AutoCad DXF	image/gif	*.gif images
application/x-autocad	AutoCad DXF	image/jpeg	*.jpeg, *.jpg images
application/x-dxf	AutoCad DXF	image/png	*.png images
application/acad	AutoCad DWG	image/tiff	*.tiff, *.tif images
application/x-shockwave-flash	Shockwave 3D	image/bmp	*.bmp images
model/x3d+vrml	X3D		

C.7 Vegetation module

Code list of the *SolitaryVegetationObject* attribute *class*			
http://www.sig3d.org/codelists/standard/vegetation/2.0/SolitaryVegetationObject_class.xml			
1000	shrub	1060	coniferous tree
1010	low plants	1070	decidous tree
1020	medium high plants	1080	bushes
1030	high plants	1090	aquatic plants
1040	grasses	1100	climber
1050	ferns	9999	unknown

Code list of the *SolitaryVegetationObject* attributes *function* and *usage*
http://www.sig3d.org/codelists/standard/vegetation/2.0/SolitaryVegetationObject_function.xml http://www.sig3d.org/codelists/standard/vegetation/2.0/SolitaryVegetationObject_usage.xml
Code lists are identically with *SolitaryVegetationObject class*

Code list of the *PlantCover* attribute *class*			
http://www.sig3d.org/codelists/standard/vegetation/2.0/PlantCover_class.xml			
1010	Lemnetea	1280	Arrhenatheretea
1020	Asplenietea rupestris	1290	Molinio-Juncetea
1030	Adiantetea	1300	Scheuchzerio-Caricetea fuscae azidophile
1040	Thlaspietea rotundifolii	1310	Festuco-Brometea
1050	Crithmo-Limonietea	1320	Elyno-Seslerietea
1060	Ammophietea	1330	Caricetea curvulae azidophile
1070	Cakiletea maritimae halophile	1340	Calluno-Ulicetea
1080	Secalinetea	1350	Oxycocco-Sphagnetea
1090	Chenopodietea	1360	Salicetea purpureae
1100	Onopordetea	1370	Betulo-Adenostyletea
1110	Epilobietea angustifolii	1380	Alnetea glutinosae
1120	Bidentetea tripartiti	1390	Erico-Pinetea
1130	Zoosteretea marinae halophile	1400	Vaccinio-Piceetea
1140	Ruppietea maritimae	1410	Quercetea robori-petraeae
1150	Potametea haftende	1420	Querco-Fagetea
1160	Litorelletea	1430	Crithmo-Staticetea
1170	Plantaginetea majoris	1440	Tuberarietea guttati
1180	Isoeto-Nanojuncetea	1450	Juncetea maritimae
1190	Montino-Cardaminetea	1460	Thero-Brachypodietea
1200	Corynephoretea	1470	Ononido-Rosmarinetea
1210	Asteretea tripolium	1480	Nerio-Tamaricetea
1220	Salicornietea	1490	Pegano-Salsoletea
1230	Juncetea maritimi	1500	Cisto-Lavanduletea
1240	Phragmitetea	1510	Quercetea ilicis
1250	Spartinetea	1520	Populetea albae
1260	Sedo-Scleranthetea	9999	unknown
1270	Salicetea herbaceae		

Code list of the *PlantCover* attributes *function* and *usage*
http://www.sig3d.org/codelists/standard/vegetation/2.0/PlantCover_function.xml http://www.sig3d.org/codelists/standard/vegetation/2.0/PlantCover_usage.xml
Code lists are identically with *PlantCover class*

Code list of the *SolitaryVegetationObject* attribute *species* (excerpt)			
http://www.sig3d.org/codelists/standard/vegetation/2.0/SolitaryVegetationObject_species.xml			
1640	Abies alba	1790	Acer circinatum
1650	Abies cephalonica	1800	Acer Davidii
1660	Abies concolor	1810	Acer ginnala Maxim
1670	Abies grandis	1820	Acer grosserii
1680	Abies homolepsis	1830	Acer monspessulanum
1690	Abies koreana	1840	Acer negundo
1700	Abies lasiocarpa	1850	Acer palmatum
1710	Abies nordmanniana	1860	Acer platanoides
1720	Abies pinsapo	1870	Acer platanoides 'Crimson King'
1730	Abies procera	1880	Acer pseudoplatanus
1740	Abies procera 'Glauca'	1890	Acer rubrum
1750	Abies veitchii	1900	Acer saccharinum
1760	Acer campéstre	1910	Acer saccharum Marsch
1770	Acer capillipes	1920	Acer tartaricum
1780	Acer cappadocicum	…	…

C.8 Transportation module

Code list of the *AuxiliaryTrafficArea* attribute *function*			
http://www.sig3d.org/codelists/standard/transportation/2.0/AuxiliaryTrafficArea_function.xml			
1000	soft shoulder	1300	traffic island
1010	hard shoulder	1400	bank
1020	green area	1410	embankment, dike
1030	middle lane	1420	railroad embankment
1040	lay by	1430	noise protection
1100	parking bay	1440	noise protection wall
1200	ditch	1500	noise guard bar
1210	drainage	1600	towpath
1220	kerbstone	1700	others
1230	flower tub		

Code list of the *TrafficArea* attribute *function*			
http://www.sig3d.org/codelists/standard/transportation/2.0/TrafficArea_function.xml			
1	driving_lane	20	crosswalk
2	footpath	21	barrier
3	cyclepath	22	stairs
4	combined foot-/cyclepath	23	escalator
5	square	24	filtering lane
6	car_park	25	airport_runway
7	parking_lay_by	26	airport_taxiway
8	rail	27	airport_apron
9	rail_road_combined	28	airport_heliport
10	drainage	29	airport_runway_marking
11	road marking	30	green spaces
12	road_marking_direction	31	recreation
13	road_marking_lane	32	bus_lay_by
14	road_marking_restricted	33	motorway
15	road_marking_crosswalk	34	motorway_entry
16	road_marking_stop	35	motorway_exit
17	road_marking_other	36	motorway_emergency lane
18	overhead wire (trolley)	37	private_area
19	train platform	9999	unknown

Code list of the *TrafficArea* attribute *usage*			
http://www.sig3d.org/codelists/standard/transportation/2.0/TrafficArea_usage.xml			
1	pedestrian	9	boat, ferry, ship
2	car	10	teleferic
3	truck	11	aeroplane
4	bus, taxi	12	helicopter
5	train	13	taxi
6	bicycle	14	horse
7	motorcycle	9999	unknown
8	tram, streetcar		

Code list of the *TrafficArea* and *AuxiliaryTrafficArea* attribute *surfaceMaterial*			
http://www.sig3d.org/codelists/standard/transportation/2.0/TrafficArea_surfaceMaterial.xml			
1	asphalt	8	soil
2	concrete	9	sand
3	pavement	10	grass
4	cobblestone	11	wood
5	gravel	12	steel
6	rail_with_bed	13	marble
7	rail_without_bed	9999	unknown

Code list of the *TransportationComplex* attribute *class*			
http://www.sig3d.org/codelists/standard/transportation/2.0/TransportationComplex_class.xml			
1000	private	1050	air traffic
1010	common	1060	rail traffic
1020	civil	1070	waterway
1030	military	1080	subway
1040	road traffic	1090	others

Code list of the *TransportationComplex* attributes *function* and *usage*			
http://www.sig3d.org/codelists/standard/transportation/2.0/TransportationComplex_function.xml http://www.sig3d.org/codelists/standard/transportation/2.0/TransportationComplex_usage.xml			
1000	road	1855	railway track
1010	freeway/motorway	1860	magnetic levitation train
1020	highway/national primary road	1900	railway station
1030	land road	1910	stop
1040	district road	1920	station
1050	municipal road	2000	power-wheel
1060	main through-road	2100	airport
1100	freeway interchange/ highway junction	2110	international airport
1110	junction	2120	regional airport
1200	road	2130	landing place
1210	driveway	2140	heliport
1220	footpath/footway	2150	landing place
1230	hiking trail	2160	gliding airfield
1240	bikeway/cycle-path	2170	taxiway
1250	bridleway/bridlepath	2180	apron
1260	main agricultural road	2190	runway
1270	agricultural road	2200	canal
1280	bikeway/footway	2300	harbor
1290	access road	2310	pleasure craft harbour
1300	dead-end road	2400	ferry
1400	lane	2410	car ferry
1410	lane, one direction	2420	train ferry
1420	lane, both direction	2430	ferry
1500	pedestrian zone	2500	landing stage
1600	place	2600	waterway I order
1610	parking area	2610	navigable river
1620	marketplace	2620	inland navigation waterway 0
1700	service area	2621	inland navigation waterway 0
1800	rail transport	2622	inland navigation waterway I
1805	rail	2623	inland navigation waterway II
1810	urban/city train	2624	inland navigation waterway III

1815	city railway	2625	inland navigation waterway IV
1820	tram	2626	inland navigation waterway V
1825	subway	2627	inland navigation waterway VI
1830	funicular/mountain railway	2628	inland navigation waterway VII
1835	mountain railway	2630	maritime navigation
1840	chairlift	2640	navigable lake
1845	ski-lift/ski tow lift	2700	others
1850	suspension railway		

C.9 WaterBody module

Code list of the *WaterBody* attribute *class*			
http://www.sig3d.org/codelists/standard/waterbody/2.0/WaterBody_class.xml			
1000	sea	1140	flooded land
1010	tidal waterbody	1150	artificial waterbody
1020	watercourse	1160	aqueduct
1030	river / stream	1170	canal
1040	ditch	1180	port basin
1050	spring / water hole	1190	reservior
1060	lake / pont	1200	excavation pont
1070	bayou	1210	moat
1080	body of standing water	1220	pool
1090	waterfall	1230	fountain
1100	rapids	1240	well
1110	swamp	1250	cistern
1120	sinkhole (karst)	1260	fish ladder
1130	ephemeral watercourse	9999	unknown

Codelist of the *WaterBody* attribute *function*			
http://www.sig3d.org/codelists/standard/waterbody/2.0/WaterBody_function.xml			
1000	nature-sanctuary	1090	public swimming
1010	protected waterbody	1100	public fountain
1020	reservoir	1110	private waterbody
1030	retention waterbody	1120	irrigation waterbody
1040	flood plain waterbody	1130	watering place
1050	waterway	1140	industrial waterbody
1060	habor waterbody	1150	waterbody for fire-fighting
1070	sluice waterbody	9999	unknown
1080	sewage system		

Codelist of the *WaterBody* attribute *usage*			
http://www.sig3d.org/codelists/standard/waterbody/2.0/WaterBody_usage.xml			
1000	sanctuary	1110	industrial / craft water supply
1010	recreation / sports	1120	military use
1020	drinking water supply	1130	mining / excavation
1030	hydroelectric water supply	1140	irrigation water supply
1040	ocean shipping	1150	fishing water
1050	inland shipping	1160	fish farm
1060	sewer	1170	archaeological site
1070	port	1180	water protection area
1080	anchorage	1190	abandoned
1090	public use	9999	unknown
1100	private use		

Code list of the *WaterSurface* attribute *waterLevel*			
http://www.sig3d.org/codelists/standard/waterbody/2.0/WaterSurface_waterLevel.xml			
1000	MSL - Mean Sea Level	1090	Hundred Year Flood
1010	LAT - Lowest Astronomical Tide	1100	highest known water level
1020	National Water Level	1110	critical low-water level

1030	Mean High Tide (related to National Waterlevel)	1120	lowest known water level
1040	Extreme High Tide (related to National Waterlevel)	1130	Established Line of Navigability
1050	Mean Low Tide (related to National Waterlevel)	1140	Minimum Limit of Navigability
1060	Extreme Low Tide (related to National Waterlevel)	1150	Maximum Limit of Navigability
1070	Mean Water Level (watercourse)	9999	unknown
1080	critical high-water level		

C.10 CityObjectGroup module

Code list of the *CityObjectGroup* attribute *class*			
http://www.sig3d.org/codelists/standard/cityobjectgroup/2.0/CityObjectGroup_class.xml			
1000	building separation	2000	assembly

Code list of the *CityObjectGroup* attributes *function* and *usage*			
http://www.sig3d.org/codelists/standard/cityobjectgroup/2.0/CityObjectGroup_function.xml http://www.sig3d.org/codelists/standard/cityobjectgroup/2.0/CityObjectGroup_usage.xml			
1000	lod1Storey	1020	lod3Storey
1010	lod2Storey	1030	lod4Storey

Annex D
(informative)

Overview of employed GML3 geometry classes

Abstract GML classes referenced in CityGML	GML subclass actually used in CityGML
_Geometry	
_Solid	Solid (boundary is restricted to OrientableSurfaces, TexturedSurfaces, Polygons or CompositeSurfaces)
	CompositeSolid
_Surface	Polygon (with holes, modelled by _Rings. The boundary is restricted to LineStrings or CompositeCurves)
	OrientableSurface
	TexturedSurface (defined in CityGML's *TexturedSurface* module, not in GML. This modelling approach has been marked deprecated)
	CompositeSurface (members are restricted to OrientableSurfaces, TexturedSurfaces, Polygons or CompositeSurfaces)
	TriangulatedSurface
	TIN
_Curve	LineString
	CompositeCurve (members are restricted to LineStrings or CompositeCurves)
	Point
_Coverage	RectifiedGridCoverage
_AbstractGeometricAggregate	MultiSolid
	MultiSurface (members are restricted to OrientableSurfaces, TexturedSurfaces, Polygons or CompositeSurfaces)
	MultiCurve (members are restricted to LineStrings or CompositeCurves)
	MultiPoint
	GeometricComplex (restricted to connected, linear networks)

Annex E
(informative)

Overview of the assignment of features to LODs

The following table lists all feature types of CityGML. For each feature type, all non-spatial and spatial properties are given including their type. Each feature is assigned a range of LOD in which it may occur, and for each property the LOD in which it represents the feature is stated. Each feature type's name is preceded by a prefix indicating the CityGML module defining the feature type. For a list of prefixes and associated CityGML modules, please refer to chapter 4.3 and chapter 7.

Feature Class	Property	Type	LOD
CityGML Core module			
core:CityModelType			0 – 4
	cityObjectMember	gml:FeaturePropertyType	0 – 4
	app:appearanceMember	app:AppearancePropertyType	0 – 4
	_GenericApplicationPropertyOfCityModel	xs:anyType	0 – 4
core:AbstractCityObjectType			0 – 4
	creationDate	xs:date	0 – 4
	terminationDate	xs:date	0 – 4
	externalReference	core:ExternalReferenceType	0 – 4
	generalizesTo	core:GeneralizationRelationType	0 – 4
	relativeToTerrain	core:RelativeToTerrainType	0 – 4
	relativeToWater	core:RelativeToWaterType	0 – 4
	app:appearance	app:AppearancePropertyType	0 – 4
	gen:_genericAttribute	gen:AbstractGenericAttributeType	0 – 4
	_GenericApplicationPropertyOfCityObject	xs:anyType	0 – 4
core:AbstractSiteType			0 – 4
	_GenericApplicationPropertyOfSite	xs:anyType	0 – 4
core:AddressType			0 – 4
	xalAddress	core:xalAddressPropertyType	0 – 4
	multiPoint	gml:MultiPointPropertyType	0 – 4
	_GenericApplicationPropertyOfAddress	xs:anyType	0 – 4
Appearance module			
app:AppearanceType			0 – 4
	theme	xs:string	0 – 4
	surfaceDataMember	app:SurfaceDataPropertyType	0 – 4
	_GenericApplicationPropertyOfAppearance	xs:anyType	0 – 4
app:AbstractSurfaceDataType			0 – 4
	isFront	xs:Boolean	0 – 4
	_GenericApplicationPropertyOfSurfaceData	xs:anyType	0 – 4
app:AbstractTextureType			0 – 4
	imageURI	xs:anyURI	0 – 4
	mimeType	gml:CodeType	0 – 4
	textureType	app:TextureTypeType	0 – 4
	wrapMode	app:WrapModeType	0 – 4
	borderColor	app:ColorPlusOpacity	0 – 4
	_GenericApplicationPropertyOfTexture	xs:anyType	0 – 4
app:ParameterizedTextureType			0 – 4
	target	app:TextureAssociationType	0 – 4
	_GenericApplicationPropertyOfParameterizedTexture	xs:anyType	0 – 4

app:GeoreferencedTextureType			0 – 4
	preferWorldFile	xs:boolean	0 – 4
	referencePoint	gml:PointPropertyType	0 – 4
	orientation	core:TransformationMatrix2x2Type	0 – 4
	target	xs:anyURI	0 – 4
	_GenericApplicationPropertyOfGeoreferencedTexture	xs:anyType	0 – 4
app:X3DMaterialType			0 – 4
	ambientIntensity	core:doubleBetween0and1	0 – 4
	diffuseColor	app:Color	0 – 4
	emissiveColor	app:Color	0 – 4
	specularColor	app:Color	0 – 4
	shininess	core:doubleBetween0and1	0 – 4
	transparency	core:doubleBetween0and1	0 – 4
	isSmooth	xs:boolean	0 – 4
	target	xs:anyURI	0 – 4
	_GenericApplicationPropertyOfX3DMaterial	xs:anyType	0 – 4
Building module			
bldg:AbstractBuildingType			0 – 4
	class	gml:CodeType	0 – 4
	function	gml:CodeType	0 – 4
	usage	gml:CodeType	0 – 4
	yearOfConstruction	xs:gYear	0 – 4
	yearOfDemolition	xs:gYear	0 – 4
	roofType	gml:CodeType	0 – 4
	measuredHeight	gml:LengthType	0 – 4
	storeysAboveGround	xs:nonNegativeInteger	0 – 4
	storeysBelowGround	xs:nonNegativeInteger	0 – 4
	storeyHeightsAboveGround	gml:MeasureOrNullListType	0 – 4
	storeyHeightsBelowGround	gml:MeasureOrNullListType	0 – 4
	lod0FootPrint	gml:MultiSurfacePropertyType	0
	lod0RoofEdge	gml:MultiSurfacePropertyType	0
	lod1Solid	gml:SolidPropertyType	1
	lod1MultiSurface	gml:MultiSurfacePropertyType	1
	lod1TerrainIntersection	gml:MultiCurvePropertyType	1
	lod2Solid	gml:SolidPropertyType	2
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod2MultiCurve	gml:MultiCurvePropertyType	2
	lod2TerrainIntersection	gml:MultiCurvePropertyType	2
	outerBuildingInstallation	bldg:BuildingInstallationPropertyType	2 – 4
	interiorBuildingInstallation	bldg:IntBuildingInstallationPropertyType	4
	boundedBy	bldg:BoundarySurfacePropertyType	2 – 4
	lod3Solid	gml:SolidPropertyType	3
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod3MultiCurve	gml:MultiCurvePropertyType	3
	lod3TerrainIntersection	gml:MultiCurvePropertyType	3
	lod4Solid	gml:SolidPropertyType	4
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	lod4MultiCurve	gml:MultiCurvePropertyType	4
	lod4TerrainIntersection	gml:MultiCurvePropertyType	4
	interiorRoom	bldg:InteriorRoomPropertyType	4
	consistsOfBuildingPart	bldg:BuildingPartPropertyType	0 – 4
	address	core:AddressPropertyType	0 – 4
	_GenericApplicationPropertyOfAbstractBuilding	xs:anyType	0 – 4
bldg:BuildingType			0 – 4
	_GenericApplicationPropertyOfBuiding	xs:anyType	0 – 4
bldg:BuildingPartType			0 – 4
	_GenericApplicationPropertyOfBuidingPart	xs:anyType	0 – 4
bldg:BuildingInstallationType			2 – 4
	class	gml:CodeType	2 – 4
	function	gml:CodeType	2 – 4

	usage	gml:CodeType	2 – 4
	lod2Geometry	gml:GeometryPropertyType	2
	lod3Geometry	gml:GeometryPropertyType	3
	lod4Geometry	gml:GeometryPropertyType	4
	lod2ImplicitRepresentation	core:ImplicitRepresentationPropertyType	2
	lod3ImplicitRepresentation	core:ImplicitRepresentationPropertyType	3
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	boundedBy	bldg:BoundarySurfacePropertyType	2 – 4
	_GenericApplicationPropertyOfBuidingInstallation	xs:anyType	2 – 4
bldg:IntBuildingInstallationType			4
	class	gml:CodeType	4
	function	gml:CodeType	4
	usage	gml:CodeType	4
	lod4Geometry	gml:GeometryPropertyType	4
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	boundedBy	bldg:BoundarySurfacePropertyType	4
	_GenericApplicationPropertyOfIntBuidingInstallation	xs:anyType	4
bldg:AbstractBoundarySurfaceType			2 – 4
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	opening	bldg:OpeningPropertyType	3 – 4
	_GenericApplicationPropertyOfBoundarySurface	xs:anyType	2 – 4
bldg:RoofSurfaceType			2 – 4
	_GenericApplicationPropertyOfRoofSurface	xs:anyType	2 – 4
bldg:WallSurfaceType			2 – 4
	_GenericApplicationPropertyOfWallSurface	xs:anyType	2 – 4
bldg:OuterCeilingSurfaceType			2 – 4
	_GenericApplicationPropertyOfOuterCeilingSurface	xs:anyType	2 – 4
bldg:OuterFloorSurfaceType			2 – 4
	_GenericApplicationPropertyOfOuterFloorSurface	xs:anyType	2 – 4
bldg:GroundSurfaceType			2 – 4
	_GenericApplicationPropertyOfGroundSurface	xs:anyType	2 – 4
bldg:ClosureSurfaceType			2 – 4
	_GenericApplicationPropertyOfClosureSurface	xs:anyType	2 – 4
bldg:FloorSurfaceType			4
	_GenericApplicationPropertyOfFloorSurface	xs:anyType	4
bldg:InteriorWallSurfaceType			4
	_GenericApplicationPropertyOfInteriorWallSurface	xs:anyType	4
bldg:CeilingSurfaceType			4
	_GenericApplicationPropertyOfCeilingSurface	xs:anyType	4
bldg:AbstractOpeningType			3 – 4
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	lod3ImplicitRepresentation	core:ImplicitRepresentationPropertyType	3
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	_GenericApplicationPropertyOfOpening	xs:anyType	3 – 4
bldg:WindowType			3 – 4
	_GenericApplicationPropertyOfWindow	xs:anyType	3 – 4
bldg:DoorType			3 – 4

	address	core:AddressPropertyType	3 – 4
	_GenericApplicationPropertyOfDoor	xs:anyType	3 – 4
bldg:RoomType			4
	class	gml:CodeType	4
	function	gml:CodeType	4
	usage	gml:CodeType	4
	lod4Solid	gml:SolidPropertyType	4
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	boundedBy	bldg:BoundarySurfacePropertyType	4
	interiorFurniture	bldg:InteriorFurniturePropertyType	4
	roomInstallation	bldg:IntBuildingInstallationPropertyType	4
	_GenericApplicationPropertyOfRoom	xs:anyType	4
bldg:BuildingFurnitureType			4
	class	gml:CodeType	4
	function	gml:CodeType	4
	usage	gml:CodeType	4
	lod4Geometry	gml:GeometryPropertyType	4
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	_GenericApplicationPropertyOfBuildingFurniture	xs:anyType	4
Bridge module			
brid:AbstractBridgeType			1 – 4
	class	gml:CodeType	1 – 4
	function	gml:CodeType	1 – 4
	usage	gml:CodeType	1 – 4
	yearOfConstruction	xs:gYear	1 – 4
	yearOfDemolition	xs:gYear	1 – 4
	isMovable	xs:boolean	1 – 4
	lod1Solid	gml:SolidPropertyType	1
	lod1MultiSurface	gml:MultiSurfacePropertyType	1
	lod1TerrainIntersection	gml:MultiCurvePropertyType	1
	lod2Solid	gml:SolidPropertyType	2
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod2MultiCurve	gml:MultiCurvePropertyType	2
	lod2TerrainIntersection	gml:MultiCurvePropertyType	2
	outerBridgeConstruction	brid:BridgeConstructionElementPropertyType	1 – 4
	outerBridgeInstallation	brid:BridgeInstallationPropertyType	2 – 4
	interiorBridgeInstallation	brid:IntBridgeInstallationPropertyType	4
	boundedBy	brid:BoundarySurfacePropertyType	2 – 4
	lod3Solid	gml:SolidPropertyType	3
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod3MultiCurve	gml:MultiCurvePropertyType	3
	lod3TerrainIntersection	gml:MultiCurvePropertyType	3
	lod4Solid	gml:SolidPropertyType	4
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	lod4MultiCurve	gml:MultiCurvePropertyType	4
	lod4TerrainIntersection	gml:MultiCurvePropertyType	4
	interiorBridgeRoom	brid:InteriorBridgeRoomPropertyType	4
	consistsOfBridgePart	brid:BridgePartPropertyType	1 – 4
	address	core:AddressPropertyType	1 – 4
	_GenericApplicationPropertyOfAbstractBridge	xs:anyType	1 – 4
brid:BridgeType			1 – 4
	_GenericApplicationPropertyOfBridge	xs:anyType	1 – 4
brid:BridgePartType			1 – 4
	_GenericApplicationPropertyOfBridgePart	xs:anyType	1 – 4
brid:BridgeConstructionElementType			2 – 4
	class	gml:CodeType	1 – 4
	function	gml:CodeType	1 – 4

	usage	gml:CodeType	1 – 4
	lod1Geometry	gml:GeometryPropertyType	1
	lod2Geometry	gml:GeometryPropertyType	2
	lod3Geometry	gml:GeometryPropertyType	3
	lod4Geometry	gml:GeometryPropertyType	4
	lod1TerrainIntersection	gml:MultiCurvePropertyType	1
	lod2TerrainIntersection	gml:MultiCurvePropertyType	2
	lod3TerrainIntersection	gml:MultiCurvePropertyType	3
	lod4TerrainIntersection	gml:MultiCurvePropertyType	4
	lod1ImplicitRepresentation	core:ImplicitRepresentationPropertyType	1
	lod2ImplicitRepresentation	core:ImplicitRepresentationPropertyType	2
	lod3ImplicitRepresentation	core:ImplicitRepresentationPropertyType	3
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	boundedBy	brid:BoundarySurfacePropertyType	2 – 4
	_GenericApplicationPropertyOfBridgeConstruc-tionElement	xs:anyType	2 – 4
brid:BridgeInstallationType			2 – 4
	class	gml:CodeType	2 – 4
	function	gml:CodeType	2 – 4
	usage	gml:CodeType	2 – 4
	lod2Geometry	gml:GeometryPropertyType	2
	lod3Geometry	gml:GeometryPropertyType	3
	lod4Geometry	gml:GeometryPropertyType	4
	lod2ImplicitRepresentation	core:ImplicitRepresentationPropertyType	2
	lod3ImplicitRepresentation	core:ImplicitRepresentationPropertyType	3
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	boundedBy	brid:BoundarySurfacePropertyType	2 – 4
	_GenericApplicationPropertyOfBridgeInstallation	xs:anyType	2 – 4
brid:IntBridgeInstallationType			4
	class	gml:CodeType	4
	function	gml:CodeType	4
	usage	gml:CodeType	4
	lod4Geometry	gml:GeometryPropertyType	4
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	boundedBy	brid:BoundarySurfacePropertyType	4
	_GenericApplicationPropertyOfIntBridgeInstallation	xs:anyType	4
brid:AbstractBoundarySurface Type			2 – 4
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	opening	brid:OpeningPropertyType	3 – 4
	_GenericApplicationPropertyOfBoundarySurface	xs:anyType	2 – 4
brid:RoofSurfaceType			2 – 4
	_GenericApplicationPropertyOfRoofSurface	xs:anyType	2 – 4
brid:WallSurfaceType			2 – 4
	_GenericApplicationPropertyOfWallSurface	xs:anyType	2 – 4
brid:OuterCeilingSurfaceType			2 – 4
	_GenericApplicationPropertyOfOuterCeilingSurface	xs:anyType	2 – 4
brid:OuterFloorSurfaceType			2 – 4
	_GenericApplicationPropertyOfOuterFloorSurface	xs:anyType	2 – 4
brid:GroundSurfaceType			2 – 4
	_GenericApplicationPropertyOfGroundSurface	xs:anyType	2 – 4
brid:ClosureSurfaceType			2 – 4
	_GenericApplicationPropertyOfClosureSurface	xs:anyType	2 – 4

brid:FloorSurfaceType			4
	_GenericApplicationPropertyOfFloorSurface	xs:anyType	4
brid:InteriorWallSurfaceType			4
	_GenericApplicationPropertyOfInteriorWallSurface	xs:anyType	4
brid:CeilingSurfaceType			4
	_GenericApplicationPropertyOfCeilingSurface	xs:anyType	4
brid:AbstractOpeningType			3 – 4
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	lod3ImplicitRepresentation	core:ImplicitRepresentationPropertyType	3
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	_GenericApplicationPropertyOfOpening	xs:anyType	3 – 4
brid:WindowType			3 – 4
	_GenericApplicationPropertyOfWindow	xs:anyType	3 – 4
brid:DoorType			3 – 4
	address	core:AddressPropertyType	3 – 4
	_GenericApplicationPropertyOfDoor	xs:anyType	3 – 4
brid: BridgeRoomType			4
	class	gml:CodeType	4
	function	gml:CodeType	4
	usage	gml:CodeType	4
	lod4Solid	gml:SolidPropertyType	4
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	boundedBy	brid:BoundarySurfacePropertyType	4
	interiorFurniture	brid:InteriorFurniturePropertyType	4
	bridgeRoomInstallation	brid:IntBridgeInstallationPropertyType	4
	_GenericApplicationPropertyOfBridgeRoom	xs:anyType	4
brid: BridgeFurnitureType			4
	class	gml:CodeType	4
	function	gml:CodeType	4
	usage	gml:CodeType	4
	lod4Geometry	gml:GeometryPropertyType	4
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	_GenericApplicationPropertyOfBridgeFurniture	xs:anyType	4
Relief module			
dem:ReliefFeatureType			0 – 4
	lod	core:integerBetween0and4	0 – 4
	reliefComponent	dem:ReliefComponentPropertyType	0 – 4
	_GenericApplicationPropertyOfReliefFeature	xs:anyType	0 – 4
dem:AbstractReliefComponentType			0 – 4
	lod	core:integerBetween0and4	0 – 4
	extent	gml:PolygonPropertyType	0 – 4
	_GenericApplicationPropertyOfReliefComponent	xs:anyType	0 – 4
dem:TINReliefType			0 – 4
	tin	dem:tinPropertyType	0 – 4
	_GenericApplicationPropertyOfTINRelief	xs:anyType	0 – 4
dem:RasterReliefType			0 – 4
	grid	dem:gridPropertyType	0 – 4
	_GenericApplicationPropertyOfRasterRelief	xs:anyType	0 – 4
dem:MassPointReliefType			0 – 4
	reliefPoints	gml:MultiPointPropertyType	0 – 4

	_GenericApplicationPropertyOfMassPointRelief	xs:anyType	0 – 4
dem:BreakLineReliefType			0 – 4
	ridgeOrValleyLines	gml:MultiCurvePropertyType	0 – 4
	breaklines	gml:MultiCurvePropertyType	0 – 4
	_GenericApplicationPropertyOfBreakLineRelief	xs:anyType	0 – 4
CityFurniture module			
frn:CityFurnitureType			1 – 4
	class	gml:CodeType	1 – 4
	function	gml:CodeType	1 – 4
	usage	gml:CodeType	1 – 4
	lod1Geometry	gml:GeometryPropertyType	1
	lod2Geometry	gml:GeometryPropertyType	2
	lod3Geometry	gml:GeometryPropertyType	3
	lod4Geometry	gml:GeometryPropertyType	4
	lod1TerrainIntersection	gml:MultiCurvePropertyType	1
	lod2TerrainIntersection	gml:MultiCurvePropertyType	2
	lod3TerrainIntersection	gml:MultiCurvePropertyType	3
	lod4TerrainIntersection	gml:MultiCurvePropertyType	4
	lod1ImplicitRepresentation	core:ImplicitRepresentationPropertyType	1
	lod2ImplicitRepresentation	core:ImplicitRepresentationPropertyType	2
	lod3ImplicitRepresentation	core:ImplicitRepresentationPropertyType	3
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	_GenericApplicationPropertyOfCityFurniture	xs:anyType	1 – 4
CityObjectGroup module			
grp:CityObjectGroupType			0 – 4
	class	gml:CodeType	0 – 4
	function	gml:CodeType	0 – 4
	usage	gml:CodeType	0 – 4
	groupMember	grp:CityObjectGroupMemberType	0 – 4
	parent	grp:CityObjectGroupParentType	0 – 4
	geometry	gml:GeometryPropertyType	0 – 4
	_GenericApplicationPropertyOfCityObjectGroup	xs:anyType	0 – 4
Generics module			
gen:GenericCityObjectType			0 – 4
	class	gml:CodeType	0 – 4
	function	gml:CodeType	0 – 4
	usage	gml:CodeType	0 – 4
	lod0Geometry	gml:GeometryPropertyType	0
	lod1Geometry	gml:GeometryPropertyType	1
	lod2Geometry	gml:GeometryPropertyType	2
	lod3Geometry	gml:GeometryPropertyType	3
	lod4Geometry	gml:GeometryPropertyType	4
	lod0TerrainIntersection	gml:MultiCurvePropertyType	0
	lod1TerrainIntersection	gml:MultiCurvePropertyType	1
	lod2TerrainIntersection	gml:MultiCurvePropertyType	2
	lod3TerrainIntersection	gml:MultiCurvePropertyType	3
	lod4TerrainIntersection	gml:MultiCurvePropertyType	4
	lod0ImplicitRepresentation	core:ImplicitRepresentationPropertyType	0
	lod1ImplicitRepresentation	core:ImplicitRepresentationPropertyType	1
	lod2ImplicitRepresentation	core:ImplicitRepresentationPropertyType	2
	lod3ImplicitRepresentation	core:ImplicitRepresentationPropertyType	3
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
LandUse module			
luse:LandUseType			0 – 4
	class	gml:CodeType	0 – 4
	function	gml:CodeType	0 – 4
	usage	gml:CodeType	0 – 4
	lod0MultiSurface	gml:MultiSurfacePropertyType	0
	lod1MultiSurface	gml:MultiSurfacePropertyType	1
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod3MultiSurface	gml:MultiSurfacePropertyType	3

	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	_GenericApplicationPropertyOfLandUse	xs:anyType	0 – 4
Transportation module			
tran:AbstractTransportationObjectType			0 – 4
	_GenericApplicationPropertyOfTransportationObject	xs:anyType	0 – 4
tran:TransportationComplexType			0 – 4
	class	gml:CodeType	0 – 4
	function	gml:CodeType	0 – 4
	usage	gml:CodeType	0 – 4
	trafficArea	tran:TrafficAreaPropertyType	0 – 4
	auxilaryTrafficArea	tran:AuxilaryTrafficAreaPropertyType	0 – 4
	lod0Network	gml:GeometricComplexPropertyType	0
	lod1MultiSurface	gml:MultiSurfacePropertyType	1
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	_GenericApplicationPropertyOfTransportationComplex	xs:anyType	0 – 4
tran:TrafficAreaType			1 – 4
	class	gml:CodeType	1 – 4
	function	gml:CodeType	1 – 4
	usage	gml:CodeType	1 – 4
	surfaceMaterial	gml:CodeType	1 – 4
	lod1MultiSurface	gml:MultiSurfacePropertyType	1
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	_GenericApplicationPropertyOfTrafficArea	xs:anyType	1 – 4
tran:AuxillaryTrafficAreaType			1 – 4
	class	gml:CodeType	1 – 4
	function	gml:CodeType	1 – 4
	usage	gml:CodeType	1 – 4
	surfaceMaterial	gml:CodeType	1 – 4
	lod1MultiSurface	gml:MultiSurfacePropertyType	1
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	_GenericApplicationPropertyOfAuxiliaryTrafficArea	xs:anyType	1 – 4
tran:TrackType			1 – 4
	_GenericApplicationPropertyOfTrack	xs:anyType	1 – 4
tran:RoadType			1 – 4
	_GenericApplicationPropertyOfRoad	xs:anyType	1 – 4
tran:RailwayType			1 – 4
	_GenericApplicationPropertyOfRailway	xs:anyType	1 – 4
tran:SquareType			1 – 4
	_GenericApplicationPropertyOfSquare	xs:anyType	1 – 4
Tunnel module			
tun:AbstractTunnelType			1 – 4
	class	gml:CodeType	1 – 4
	function	gml:CodeType	1 – 4
	usage	gml:CodeType	1 – 4
	yearOfConstruction	xs:gYear	1 – 4
	yearOfDemolition	xs:gYear	1 – 4
	lod1Solid	gml:SolidPropertyType	1
	lod1MultiSurface	gml:MultiSurfacePropertyType	1
	lod1TerrainIntersection	gml:MultiCurvePropertyType	1

	lod2Solid	gml:SolidPropertyType	2
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod2MultiCurve	gml:MultiCurvePropertyType	2
	lod2TerrainIntersection	gml:MultiCurvePropertyType	2
	outerTunnelInstallation	tun:TunnelInstallationPropertyType	2 – 4
	interiorTunnelInstallation	tun:IntTunnelInstallationPropertyType	4
	boundedBy	tun:BoundarySurfacePropertyType	2 – 4
	lod3Solid	gml:SolidPropertyType	3
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod3MultiCurve	gml:MultiCurvePropertyType	3
	lod3TerrainIntersection	gml:MultiCurvePropertyType	3
	lod4Solid	gml:SolidPropertyType	4
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	lod4MultiCurve	gml:MultiCurvePropertyType	4
	lod4TerrainIntersection	gml:MultiCurvePropertyType	4
	interiorHollowSpace	tun:InteriorHollowSpacePropertyType	4
	consistsOfTunnelPart	tun:TunnelPartPropertyType	1 – 4
	_GenericApplicationPropertyOfAbstractTunnel	xs:anyType	1 – 4
tun:TunnelType			1 – 4
	_GenericApplicationPropertyOfTunnel	xs:anyType	1 – 4
tun:TunnelPartType			1 – 4
	_GenericApplicationPropertyOfTunnelPart	xs:anyType	1 – 4
tun:TunnelInstallationType			2 – 4
	class	gml:CodeType	2 – 4
	function	gml:CodeType	2 – 4
	usage	gml:CodeType	2 – 4
	lod2Geometry	gml:GeometryPropertyType	2
	lod3Geometry	gml:GeometryPropertyType	3
	lod4Geometry	gml:GeometryPropertyType	4
	lod2ImplicitRepresentation	core:ImplicitRepresentationPropertyType	2
	lod3ImplicitRepresentation	core:ImplicitRepresentationPropertyType	3
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	boundedBy	tun:BoundarySurfacePropertyType	2 – 4
	_GenericApplicationPropertyOfTunnelInstallation	xs:anyType	2 – 4
tun:IntTunnelInstallationType			4
	class	gml:CodeType	4
	function	gml:CodeType	4
	usage	gml:CodeType	4
	lod4Geometry	gml:GeometryPropertyType	4
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	boundedBy	tun:BoundarySurfacePropertyType	4
	_GenericApplicationPropertyOfIntTunnelInstallation	xs:anyType	4
tun:AbstractBoundarySurfaceType			2 – 4
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	opening	tun:OpeningPropertyType	3 – 4
	_GenericApplicationPropertyOfBoundarySurface	xs:anyType	2 – 4
tun:RoofSurfaceType			2 – 4
	_GenericApplicationPropertyOfRoofSurface	xs:anyType	2 – 4
tun:WallSurfaceType			2 – 4
	_GenericApplicationPropertyOfWallSurface	xs:anyType	2 – 4
tun:OuterCeilingSurfaceType			2 – 4
	_GenericApplicationPropertyOfOuterCeilingSurface	xs:anyType	2 – 4
tun:OuterFloorSurfaceType			2 – 4

	_GenericApplicationPropertyOfOuterFloorSurface	xs:anyType	2 – 4
tun:GroundSurfaceType			2 – 4
	_GenericApplicationPropertyOfGroundSurface	xs:anyType	2 – 4
tun:ClosureSurfaceType			2 – 4
	_GenericApplicationPropertyOfClosureSurface	xs:anyType	2 – 4
tun:FloorSurfaceType			4
	_GenericApplicationPropertyOfFloorSurface	xs:anyType	4
tun:InteriorWallSurfaceType			4
	_GenericApplicationPropertyOfInteriorWallSurface	xs:anyType	4
tun:CeilingSurfaceType			4
	_GenericApplicationPropertyOfCeilingSurface	xs:anyType	4
tun:AbstractOpeningType			3 – 4
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	lod3ImplicitRepresentation	core:ImplicitRepresentationPropertyType	3
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	_GenericApplicationPropertyOfOpening	xs:anyType	3 – 4
tun:WindowType			3 – 4
	_GenericApplicationPropertyOfWindow	xs:anyType	3 – 4
tun:DoorType			3 – 4
	address	core:AddressPropertyType	3 – 4
	_GenericApplicationPropertyOfDoor	xs:anyType	3 – 4
tun:HollowSpaceType			4
	class	gml:CodeType	4
	function	gml:CodeType	4
	usage	gml:CodeType	4
	lod4Solid	gml:SolidPropertyType	4
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	boundedBy	tun:BoundarySurfacePropertyType	4
	interiorFurniture	tun:InteriorFurniturePropertyType	4
	hollowSpaceInstallation	tun:IntTunnelInstallationPropertyType	4
	_GenericApplicationPropertyOfHollowSpace	xs:anyType	4
tun:TunnelFurnitureType			4
	class	gml:CodeType	4
	function	gml:CodeType	4
	usage	gml:CodeType	4
	lod4Geometry	gml:GeometryPropertyType	4
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	_GenericApplicationPropertyOfTunnelFurniture	xs:anyType	4
Vegetation module			
veg:AbstractVegetationObjectType			1 – 4
	_GenericApplicationPropertyOfVegetationObject	xs:anyType	1 – 4
veg:PlantCoverType			1 – 4
	class	gml:CodeType	1 – 4
	function	gml:CodeType	1 – 4
	usage	gml:CodeType	1 – 4
	averageHeight	gml:LengthType	1 – 4
	lod1MultiSurface	gml:MultiSurfacePropertyType	1
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	lod1MultiSolid	gml:MultiSolidPropertyType	1

	lod2MultiSolid	gml:MultiSolidPropertyType	2
	lod3MultiSolid	gml:MultiSolidPropertyType	3
	Lod4MultiSolid	gml:MultiSolidPropertyType	4
	_GenericApplicationPropertyOfPlantCover	xs:anyType	1 – 4
veg:SolitaryVegetationObjectType			1 – 4
	class	gml:CodeType	1 – 4
	function	gml:CodeType	1 – 4
	usage	gml:CodeType	1 – 4
	species	gml:CodeType	1 – 4
	height	gml:LengthType	1 – 4
	trunkDiameter	gml:LengthType	1 – 4
	crownDiameter	gml:LengthType	1 – 4
	lod1Geometry	gml:GeometryPropertyType	1
	lod2Geometry	gml:GeometryPropertyType	2
	lod3Geometry	gml:GeometryPropertyType	3
	lod4Geometry	gml:GeometryPropertyType	4
	lod1ImplicitRepresentation	core:ImplicitRepresentationPropertyType	1
	lod2ImplicitRepresentation	core:ImplicitRepresentationPropertyType	2
	lod3ImplicitRepresentation	core:ImplicitRepresentationPropertyType	3
	lod4ImplicitRepresentation	core:ImplicitRepresentationPropertyType	4
	_GenericApplicationPropertyOfSolitaryVegetationObject	xs:anyType	1 – 4
WaterObject module			
wtr:AbstractWaterObjectType			0 – 4
	_GenericApplicationPropertyOfWaterObject	xs:anyType	0 – 4
wtr:WaterBodyType			0 – 4
	class	gml:CodeType	0 – 4
	function	gml:CodeType	0 – 4
	usage	gml:CodeType	0 – 4
	lod0MultiCurve	gml:MultiCurvePropertyType	0
	lod1MultiCurve	gml:MultiCurvePropertyType	1
	lod0MultiSurface	gml:MultiSurfacePropertyType	0
	lod1MultiSurface	gml:MultiSurfacePropertyType	1
	lod1Solid	gml:SolidPropertyType	1
	lod2Solid	gml:SolidPropertyType	2
	lod3Solid	gml:SolidPropertyType	3
	lod4Solid	gml:SolidPropertyType	4
	boundedBy	wtr:BoundedByWaterSurfacePropertyType	2 – 4
	_GenericApplicationPropertyOfWaterBody	xs:anyType	0 – 4
wtr:AbstractWaterBoundarySurfaceType			2 – 4
	lod2Surface	gml:SurfacePropertyType	2
	lod3Surface	gml:SurfacePropertyType	3
	lod4Surface	gml:SurfacePropertyType	4
	_GenericApplicationPropertyOfWaterBoundarySurface	xs:anyType	2 – 4
wtr:WaterSurfaceType			2 – 4
	waterLevel	WaterLevelType	2 – 4
	_GenericApplicationPropertyOfWaterSurface	xs:anyType	2 – 4
wtr:WaterGroundSurfaceType			2 – 4
	_GenericApplicationPropertyOfWaterGroundSurface	xs:anyType	2 – 4
wtr:WaterClosureSurfaceType			2 – 4
	_GenericApplicationPropertyOfWaterClosureSurface	xs:anyType	2 – 4

Annex F
(informative)

Changelog for CityGML 2.0

The following table lists all feature types, properties, and data types which have been added or changed for CityGML 2.0. In order to maintain backwards compatibility, none of the classes marked as being deprecated in CityGML 1.0 has been deleted.

Feature Class / Data Type	Property	New	Changed	Deleted	Description of change
CityGML Core module					
core:AbstractCityObjectType					
	relativeToTerrain	x			new attribute
	relativeToWater	x			new attribute
Appearance module					
app:AbstractTextureType					
	mimeType		x		→gml:CodeType
Building module					
bldg:AbstractBuildingType					
	class		x		→gml:CodeType
	function		x		→gml:CodeType
	usage		x		→gml:CodeType
	roofType		x		→gml:CodeType
	lod0FootPrint	x			new attribute
	lod0RoofEdge	x			new attribute
bldg:BuildingClassType				x	replaced by gml:CodeType
bldg:BuildingFunctionType				x	replaced by gml:CodeType
bldg:BuildingUsageType				x	replaced by gml:CodeType
bldg:RoofTypeType				x	replaced by gml:CodeType
bldg:BuildingInstallationType					
	class		x		→gml:CodeType
	function		x		→gml:CodeType
	usage		x		→gml:CodeType
	lod2ImplicitRepresentation	x			new attribute
	lod3ImplicitRepresentation	x			new attribute
	lod4ImplicitRepresentation	x			new attribute
	boundedBy	x			new attribute
bldg:BuildingInstallation ClassType				x	replaced by gml:CodeType
bldg:BuildingInstallation FunctionType				x	replaced by gml:CodeType
bldg:BuildingInstallation UsageType				x	replaced by gml:CodeType
bldg:IntBuildingInstallationType					
	class		x		→gml:CodeType
	function		x		→gml:CodeType
	usage		x		→gml:CodeType
	lod4ImplicitRepresentation		x		new attribute
	boundedBy		x		new attribute
bldg:IntBuildingInstallation ClassType				x	replaced by gml:CodeType
bldg:IntBuildingInstallation				x	replaced by

FunctionType					gml:CodeType
bldg:IntBuildingInstallation UsageType				x	replaced by gml:CodeType
bldg:OuterCeilingSurfaceType		x			new feature
	_GenericApplicationPropertyOfOuter CeilingSurface	x			new attribute
bldg:OuterFloorSurfaceType		x			new feature
	_GenericApplicationPropertyOfOuter FloorSurface	x			new attribute
bldg:AbstractOpeningType					
	lod3ImplicitRepresentation	x			new attribute
	lod4ImplicitRepresentation	x			new attribute
bldg:RoomType					
	class		x		→gml:CodeType
	function		x		→gml:CodeType
	usage		x		→gml:CodeType
bldg:RoomClassType				x	replaced by gml:CodeType
bldg:RoomFunctionType				x	replaced by gml:CodeType
bldg:RoomUsageType				x	replaced by gml:CodeType
bldg:BuildingFurnitureType					
	class		x		→gml:CodeType
	function		x		→gml:CodeType
	usage		x		→gml:CodeType
bldg:BuildingFurnitureClassType				x	replaced by gml:CodeType
bldg:BuildingFurnitureFunction Type				x	replaced by gml:CodeType
bldg:BuildingFurnitureUsage Type				x	replaced by gml:CodeType
Bridge module		x			New module
brid:AbstractBridgeType		x			new feature
	class	x			new attribute
	function	x			new attribute
	usage	x			new attribute
	yearOfConstruction	x			new attribute
	yearOfDemolition	x			new attribute
	isMovable	x			new attribute
	lod1Solid	x			new attribute
	lod1MultiSurface	x			new attribute
	lod1TerrainIntersection	x			new attribute
	lod2Solid	x			new attribute
	lod2MultiSurface	x			new attribute
	lod2MultiCurve	x			new attribute
	lod2TerrainIntersection	x			new attribute
	outerBridgeConstruction	x			new attribute
	outerBridgeInstallation	x			new attribute
	interiorBridgeInstallation	x			new attribute
	boundedBy	x			new attribute
	lod3Solid	x			new attribute
	lod3MultiSurface	x			new attribute
	lod3MultiCurve	x			new attribute
	lod3TerrainIntersection	x			new attribute
	lod4Solid	x			new attribute
	lod4MultiSurface	x			new attribute
	lod4MultiCurve	x			new attribute
	lod4TerrainIntersection	x			new attribute
	interiorBridgeRoom	x			new attribute
	consistsOfBridgePart	x			new attribute
	address	x			new attribute
	_GenericApplicationPropertyOfAbstract Bridge	x			new attribute
brid:BridgeType		x			new feature

	_GenericApplicationPropertyOfBridge	x			new attribute
brid:BridgePartType		x			new feature
	_GenericApplicationPropertyOfBridge Part	x			new attribute
brid:BridgeConstructionElement Type		x			new feature
	class	x			new attribute
	function	x			new attribute
	usage	x			new attribute
	lod1Geometry	x			new attribute
	lod2Geometry	x			new attribute
	lod3Geometry	x			new attribute
	lod4Geometry	x			new attribute
	lod1TerrainIntersection	x			new attribute
	lod2TerrainIntersection	x			new attribute
	lod3TerrainIntersection	x			new attribute
	lod4TerrainIntersection	x			new attribute
	lod1ImplicitRepresentation	x			new attribute
	lod2ImplicitRepresentation	x			new attribute
	lod3ImplicitRepresentation	x			new attribute
	lod4ImplicitRepresentation	x			new attribute
	boundedBy	x			new attribute
	_GenericApplicationPropertyOfBridge ConstructionElement	x			new attribute
brid:BridgeInstallationType		x			new feature
	class	x			new attribute
	function	x			new attribute
	usage	x			new attribute
	lod2Geometry	x			new attribute
	lod3Geometry	x			new attribute
	lod4Geometry	x			new attribute
	lod2ImplicitRepresentation	x			new attribute
	lod3ImplicitRepresentation	x			new attribute
	lod4ImplicitRepresentation	x			new attribute
	boundedBy	x			new attribute
	_GenericApplicationPropertyOfBridge Installation	x			new attribute
brid:IntBridgeInstallationType		x			new feature
	class	x			new attribute
	function	x			new attribute
	usage	x			new attribute
	lod4Geometry	x			new attribute
	lod4ImplicitRepresentation	x			new attribute
	boundedBy	x			new attribute
	_GenericApplicationPropertyOfInt BridgeInstallation	x			new attribute
brid:AbstractBoundarySurface Type		x			new feature
	lod2MultiSurface	x			new attribute
	lod3MultiSurface	x			new attribute
	lod4MultiSurface	x			new attribute
	opening	x			new attribute
	_GenericApplicationPropertyOf BoundarySurface	x			new attribute
brid:RoofSurfaceType		x			new feature
	_GenericApplicationPropertyOf RoofSurface	x			new attribute
brid:WallSurfaceType		x			new feature
	_GenericApplicationPropertyOf WallSurface	x			new attribute
brid:OuterCeilingSurfaceType		x			new feature
	_GenericApplicationPropertyOf OuterCeilingSurface	x			new attribute
brid:OuterFloorSurfaceType		x			new feature
	_GenericApplicationPropertyOf	x			new attribute

	OuterFloorSurface				
brid:GroundSurfaceType		x			new feature
	_GenericApplicationPropertyOf GroundSurface	x			new attribute
brid:ClosureSurfaceType		x			new feature
	_GenericApplicationPropertyOf ClosureSurface	x			new attribute
brid:FloorSurfaceType		x			new feature
	_GenericApplicationPropertyOf FloorSurface	x			new attribute
brid:InteriorWallSurfaceType		x			new feature
	_GenericApplicationPropertyOf InteriorWallSurface	x			new attribute
brid:CeilingSurfaceType		x			new feature
	_GenericApplicationPropertyOf CeilingSurface	x			new attribute
brid:AbstractOpeningType		x			new feature
	lod3MultiSurface	x			new attribute
	lod4MultiSurface	x			new attribute
	lod3ImplicitRepresentation	x			new attribute
	lod4ImplicitRepresentation	x			new attribute
	_GenericApplicationPropertyOfOpening	x			new attribute
brid:WindowType		x			new feature
	_GenericApplicationPropertyOfWindow	x			new attribute
brid:DoorType		x			new feature
	address	x			new attribute
	_GenericApplicationPropertyOfDoor	x			new attribute
brid:BridgeRoomType		x			new feature
	class	x			new attribute
	function	x			new attribute
	usage	x			new attribute
	lod4Solid	x			new attribute
	lod4MultiSurface	x			new attribute
	boundedBy	x			new attribute
	interiorFurniture	x			new attribute
	bridgeRoomInstallation	x			new attribute
	_GenericApplicationPropertyOf BridgeRoom	x			new attribute
brid:BridgeFurnitureType		x			new feature
	class	x			new attribute
	function	x			new attribute
	usage	x			new attribute
	lod4Geometry	x			new attribute
	lod4ImplicitRepresentation	x			new attribute
	_GenericApplicationPropertyOfBridge Furniture	x			new attribute
CityFurniture module					
frn:CityFurnitureType					
	class		x		→gml:CodeType
	function		x		→gml:CodeType
	usage	x			new attribute
frn:CityFurnitureClassType				x	replaced by gml:CodeType
frn:CityFurnitureFunctionType				x	replaced by gml:CodeType
CityObjectGroup module					
grp:CityObjectGroupType					
	class		x		→gml:CodeType
	function		x		→gml:CodeType
	usage		x		→gml:CodeType
Generics module					
gen:GenericCityObjectType					
	class		x		→gml:CodeType
	function		x		→gml:CodeType
	usage		x		→gml:CodeType

gen:GenericAttributeSetType		x			new data type
gen:MeasureAttributeType		x			new data type
LandUse module					
luse:LandUseType					
	class		x		→gml:CodeType
	function		x		→gml:CodeType
	usage		x		→gml:CodeType
luse:LandUseClassType				x	replaced by gml:CodeType
luse:LandUseFunctionType				x	replaced by gml:CodeType
luse:LandUseUsageType				x	replaced by gml:CodeType
Transportation module					
tran:TransportationComplex Type					
	class	x			new attribute
	function		x		→gml:CodeType
	usage		x		→gml:CodeType
tran:TransportationComplex FunctionType				x	replaced by gml:CodeType
tran:TransportationComplex UsageType				x	replaced by gml:CodeType
tran:TrafficAreaType					
	class	x			new attribute
	function		x		→gml:CodeType
	usage		x		→gml:CodeType
	surfaceMaterial		x		→gml:CodeType
tran:TrafficAreaFunctionType				x	replaced by gml:CodeType
tran:TrafficAreaUsageType				x	replaced by gml:CodeType
tran:TrafficSurfaceMaterialType				x	replaced by gml:CodeType
tran:AuxillaryTrafficAreaType					
	class	x			new attribute
	function		x		→gml:CodeType
	usage	x			new attribute
	surfaceMaterial		x		→gml:CodeType
tran:AuxiliaryTrafficArea FunctionType				x	replaced by gml:CodeType
Tunnel module		x			New module
tun:AbstractTunnelType		x			new feature
	class	x			new attribute
	function	x			new attribute
	usage	x			new attribute
	yearOfConstruction	x			new attribute
	yearOfDemolition	x			new attribute
	lod1Solid	x			new attribute
	lod1MultiSurface	x			new attribute
	lod1TerrainIntersection	x			new attribute
	lod2Solid	x			new attribute
	lod2MultiSurface	x			new attribute
	lod2MultiCurve	x			new attribute
	lod2TerrainIntersection	x			new attribute
	outerTunnelInstallation	x			new attribute
	interiorTunnelInstallation	x			new attribute
	boundedBy	x			new attribute
	lod3Solid	x			new attribute
	lod3MultiSurface	x			new attribute
	lod3MultiCurve	x			new attribute
	lod3TerrainIntersection	x			new attribute
	lod4Solid	x			new attribute
	lod4MultiSurface	x			new attribute
	lod4MultiCurve	x			new attribute

	lod4TerrainIntersection	x			new attribute
	interiorHollowSpace	x			new attribute
	consistsOfTunnelPart	x			new attribute
	_GenericApplicationPropertyOfAbstract Tunnel	x			new attribute
tun:TunnelType		x			new feature
	_GenericApplicationPropertyOfTunnel	x			new attribute
tun:TunnelPartType		x			new feature
	_GenericApplicationPropertyOfTunnel Part	x			new attribute
tun:TunnelInstallationType		x			new feature
	class	x			new attribute
	function	x			new attribute
	usage	x			new attribute
	lod2Geometry	x			new attribute
	lod3Geometry	x			new attribute
	lod4Geometry	x			new attribute
	lod2ImplicitRepresentation	x			new attribute
	lod3ImplicitRepresentation	x			new attribute
	lod4ImplicitRepresentation	x			new attribute
	boundedBy	x			new attribute
	_GenericApplicationPropertyOfTunnel Installation	x			new attribute
tun:IntTunnelInstallationType		x			new feature
	class	x			new attribute
	function	x			new attribute
	usage	x			new attribute
	lod4Geometry	x			new attribute
	lod4ImplicitRepresentation	x			new attribute
	boundedBy	x			new attribute
	_GenericApplicationPropertyOfInt TunnelInstallation	x			new attribute
tun:AbstractBoundarySurface Type		x			new feature
	lod2MultiSurface	x			new attribute
	lod3MultiSurface	x			new attribute
	lod4MultiSurface	x			new attribute
	opening	x			new attribute
	_GenericApplicationPropertyOf BoundarySurface	x			new attribute
tun:RoofSurfaceType		x			new feature
	_GenericApplicationPropertyOf RoofSurface	x			new attribute
tun:WallSurfaceType		x			new feature
	_GenericApplicationPropertyOf WallSurface	x			new attribute
tun:OuterCeilingSurfaceType		x			new feature
	_GenericApplicationPropertyOf OuterCeilingSurface	x			new attribute
tun:OuterFloorSurfaceType		x			new feature
	_GenericApplicationPropertyOf OuterFloorSurface	x			new attribute
tun:GroundSurfaceType		x			new feature
	_GenericApplicationPropertyOf GroundSurface	x			new attribute
tun:ClosureSurfaceType		x			new feature
	_GenericApplicationPropertyOf ClosureSurface	x			new attribute
tun:FloorSurfaceType		x			new feature
	_GenericApplicationPropertyOf FloorSurface	x			new attribute
tun:InteriorWallSurfaceType		x			new feature
	_GenericApplicationPropertyOf InteriorWallSurface	x			new attribute
tun:CeilingSurfaceType		x			new feature

	_GenericApplicationPropertyOf CeilingSurface	x			new attribute
tun:AbstractOpeningType		x			new feature
	lod3MultiSurface	x			new attribute
	lod4MultiSurface	x			new attribute
	lod3ImplicitRepresentation	x			new attribute
	lod4ImplicitRepresentation	x			new attribute
	_GenericApplicationPropertyOfOpening	x			new attribute
tun:WindowType		x			new feature
	_GenericApplicationPropertyOfWindow	x			new attribute
tun:DoorType		x			new feature
	_GenericApplicationPropertyOfDoor	x			new attribute
tun:HollowSpaceType		x			new feature
	class	x			new attribute
	function	x			new attribute
	usage	x			new attribute
	lod4Solid	x			new attribute
	lod4MultiSurface	x			new attribute
	boundedBy	x			new attribute
	interiorFurniture	x			new attribute
	hollowSpaceInstallation	x			new attribute
	_GenericApplicationPropertyOf HollowSpace	x			new attribute
tun:TunnelFurnitureType		x			new feature
	class	x			new attribute
	function	x			new attribute
	usage	x			new attribute
	lod4Geometry	x			new attribute
	lod4ImplicitRepresentation	x			new attribute
	_GenericApplicationPropertyOf TunnelFurniture	x			new attribute
Vegetation module					
veg:PlantCoverType					
	class		x		→gml:CodeType
	function		x		→gml:CodeType
	usage	x			new attribute
	lod4MultiSolid	x			New Representation
veg:PlantCoverClassType				x	replaced by gml:CodeType
veg:PlantCoverFunctionType				x	replaced by gml:CodeType
veg:SolitaryVegetationObject Type					
	class		x		→gml:CodeType
	function		x		→gml:CodeType
	usage	x			new attribute
	species		x		→gml:CodeType
veg:PlantClassType				x	replaced by gml:CodeType
veg:PlantFunctionType				x	replaced by gml:CodeType
veg:SpeciesType				x	replaced by gml:CodeType
WaterObject module					
wtr:WaterBodyType					
	class		x		→gml:CodeType
	function		x		→gml:CodeType
	usage		x		→gml:CodeType
wtr:WaterBodyClassType				x	replaced by gml:CodeType
wtr:WaterBodyFunctionType				x	replaced by gml:CodeType
wtr:WaterBodyUsageType				x	replaced by gml:CodeType
wtr:WaterSurfaceType					

	waterLevel		x		→gml:CodeType
wtr:WaterLevelType				x	replaced by gml:CodeType

Annex G
(informative)

Example CityGML datasets

The following chapters provide example CityGML models and excerpts of the corresponding CityGML instance documents. In chapters G.1 to G.6, a single CityGML building is subsequently illustrated and refined from a coarse LOD0 representation up to a semantically rich and geometric-topologically sound LOD4 model including the building interior. The model exemplifies concepts from the CityGML *Building* module and demonstrates the coherent modelling of semantics and geometric/topological properties. In chapter G.7, the building and its surrounding terrain are enriched with visual appearance information to show the usage of the CityGML *Appearance* module. A further example for the *Appearance* module is presented in chapter G.8 illustrating the texturing of polygons with inner holes. Finally, the use of a local engineering coordinate reference system in CityGML is illustrated in chapter G.9.

All CityGML instance documents contained in this annex are bundled with the CityGML schema package, and are available at http://schemas.opengis.net/citygml/examples/2.0/.

G.1 Example of a CityGML dataset for a building in LOD0

The first example illustrates a CityGML building in LOD0 surrounded by a terrain model represented as LOD0 Triangulated Irregular Network (*dem:TINRefief*). Both a footprint and a roof edge representation is assigned to the building (*bldg:lod0FootPrint* and *bldg:lod0RoofEdge*). The geometry is given as *gml:MultiSurface* following the conformance requirement no. 2 of the *Building* module (cf. chapter 10.3.9) according to which the height values of all 3D coordinate tuples belonging to the same surface must be identical (which results in horizontal surfaces). The footprint (painted in dark blue) is located at the lowest elevation of the building whereas the roof egde representation (painted in light cyan) is at roof level (eaves height) and hence is floating over the terrain. The building is assigned address information using the xAL address schema. The left side of Fig. 75 visualizes the LOD0 model, whereas the hierarchy of the contained semantic entities is depicted on the right.

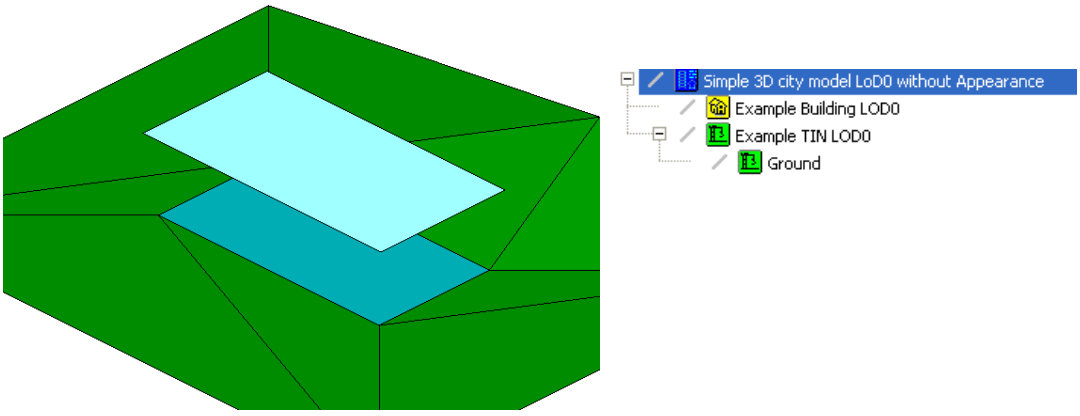

Fig. 75: Example of a CityGML building model (footprint and roof edge) and a terrain model in LOD0 (left: 3D graphic; right: model hierachy). The footprint (painted in dark blue) is located at the lowest elevation of the building whereas the roof egde representation (painted in light cyan) is at roof level (eaves height) and hence is floating over the terrain.

```
<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.opengis.net/citygml/2.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
  xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
  xsi:schemaLocation="http://www.opengis.net/citygml/building/2.0 http://schemas.opengis.net/citygml/building/2.0/building.xsd
                      http://www.opengis.net/citygml/relief/2.0 http://schemas.opengis.net/citygml/relief/2.0/relief.xsd ">
```



```
<gml:name>Simple 3D city model LOD0 without Appearance</gml:name>
<gml:boundedBy>
  <gml:Envelope srsDimension="3" srsName="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783">
    <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
    <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
  </gml:Envelope>
</gml:boundedBy>
<cityObjectMember>
  <bldg:Building gml:id="GML_7b1a5a6f-ddad-4c3d-a507-3eb9ee0a8e68">
    <gml:name>Example Building LOD0 </gml:name>
    <bldg:function
        codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_function.xml">1000</bldg:function>
    <bldg:yearOfConstruction>1985</bldg:yearOfConstruction>
    <bldg:roofType
        codeSpace=http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml">1030</bldg:roofType>
    <bldg:measuredHeight uom="#m">5.0</bldg:measuredHeight>
    <bldg:storeysAboveGround>1</bldg:storeysAboveGround>
    <bldg:storeyHeightsAboveGround uom="#m">3.0</bldg:storeyHeightsAboveGround>
    <bldg:lod0FootPrint>
      <gml:MultiSurface>
        <gml:surfaceMember>
          <gml:Polygon>
            <gml:exterior>
              <gml:LinearRing>
                <gml:posList>458875.0 5438350.0 112.0 458885.0 5438350.0 112.0 458885.0 5438355.0 112.0 458875.0
                  5438355.0 112.0 458875.0 5438350.0 112.0 </gml:posList>
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </gml:surfaceMember>
      </gml:MultiSurface>
    </bldg:lod0FootPrint>
    <bldg:lod0RoofEdge>
      <gml:MultiSurface>
        <gml:surfaceMember>
          <gml:Polygon>
            <gml:exterior>
              <gml:LinearRing>
                <gml:posList>458874.6 5438355.312347524 115.0 458874.6 5438349.687652476 115.0 458885.4 5438349.687652476
                  115.0 458885.4 5438355.312347524 115.0 458874.6 5438355.312347524 115.0 </gml:posList>
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </gml:surfaceMember>
      </gml:MultiSurface>
    </bldg:lod0RoofEdge>
    <bldg:address>
      <Address>
        <xalAddress>
          <xAL:AddressDetails>
            <xAL:Country>
              <xAL:CountryName>Germany</xAL:CountryName>
              <xAL:Locality Type="Town">
                <xAL:LocalityName>Eggenstein-Leopoldshafen</xAL:LocalityName>
                <xAL:Thoroughfare Type="Street">
                  <xAL:ThoroughfareNumber>1</xAL:ThoroughfareNumber>
                  <xAL:ThoroughfareName>Hermann-von-Helmholtz-Platz</xAL:ThoroughfareName>
                </xAL:Thoroughfare>
                <xAL:PostalCode>
                  <xAL:PostalCodeNumber>76344</xAL:PostalCodeNumber>
                </xAL:PostalCode>
              </xAL:Locality>
            </xAL:Country>
          </xAL:AddressDetails>
        </xalAddress>
        <multiPoint>
          <gml:MultiPoint>
            <gml:pointMember>
              <gml:Point>
                <gml:pos srsDimension="3">458880.0 5438352.6 112.0 </gml:pos>
              </gml:Point>
            </gml:pointMember>
          </gml:MultiPoint>
        </multiPoint>
      </Address>
    </bldg:address>
```

```
    </bldg:Building>
  </cityObjectMember>
  <cityObjectMember>
    <dem:ReliefFeature gml:id="GML_6bb30328-7599-4500-90ef-766fde6aa67b">
      <gml:name>Example TIN LOD0</gml:name>
      <dem:lod>0</dem:lod>
      <dem:reliefComponent>
        <dem:TINRelief gml:id="GUID_04D4DsNGv1MfvYu5O3lkcW">
          <gml:name>Ground</gml:name>
          <dem:lod>0</dem:lod>
          <dem:tin>
            <gml:TriangulatedSurface gml:id="ground">
              <gml:trianglePatches>
                <gml:Triangle>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:posList>458868 5438362 112 458875 5438355 112 458883 5438362 114 458868 5438362 112 </gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Triangle>
                …
              </gml:trianglePatches>
            </gml:TriangulatedSurface>
          </dem:tin>
        </dem:TINRelief>
      </dem:reliefComponent>
    </dem:ReliefFeature>
  </cityObjectMember>
</CityModel>
```

Listing 3: Excerpt from the CityGML dataset for a building in LOD0 visualised in Fig. 75.

G.2 Example of a CityGML dataset for a building in LOD1

The following Fig. 76 shows both the building and the terrain model in LOD1. The building is represented as block model. Its geometry is described by a solid (*gml:Solid*) whose exterior shell is bounded by six surfaces (*gml:Polygon*). In addition to the geometry, the building is enriched with further thematic attributes. Some of these attributes are enumerative and demonstrate the usage of the CityGML *code list* mechanism (cf. chapter 10.14). The coded attribute values are taken from the code lists proposed by the SIG 3D (cf. Annex C).

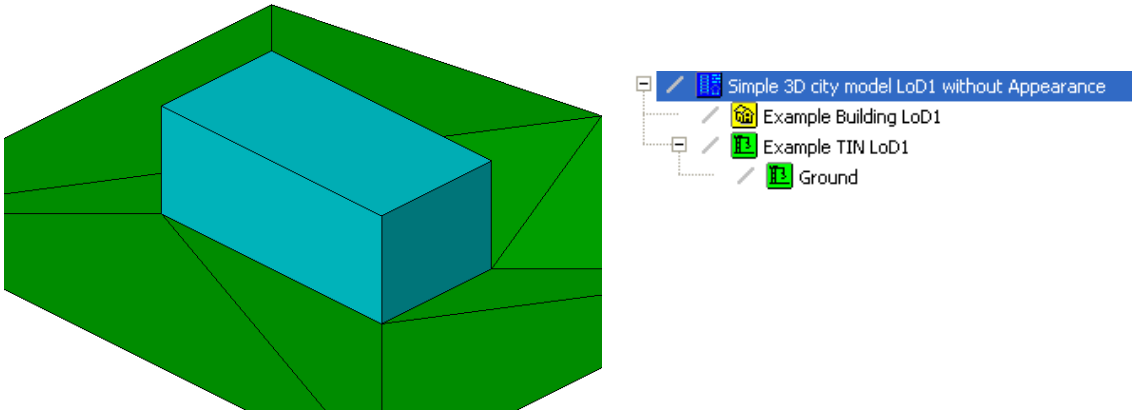

Fig. 76: Example of a CityGML building model in LOD1 (left: 3D graphic; right: model hierachy).

```
<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.opengis.net/citygml/2.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
  xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
  xsi:schemaLocation="http://www.opengis.net/citygml/building/2.0 http://schemas.opengis.net/citygml/building/2.0/building.xsd
                      http://www.opengis.net/citygml/relief/2.0 http://schemas.opengis.net/citygml/relief/2.0/relief.xsd ">
  <gml:name>Simple 3D city model LOD1 without Appearance</gml:name>
  <gml:boundedBy>
    <gml:Envelope srsDimension="3" srsName="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783">
      <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
      <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <cityObjectMember>
    <bldg:Building gml:id="GML_7b1a5a6f-ddad-4c3d-a507-3eb9ee0a8e68">
      <gml:name>Example Building LOD1 </gml:name>
      <bldg:function
          codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_function.xml">1000</bldg:function>
      <bldg:yearOfConstruction>1985</bldg:yearOfConstruction>
      <bldg:roofType
          codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml">1030</bldg:roofType>
      <bldg:measuredHeight uom="#m">5.0</bldg:measuredHeight>
      <bldg:storeysAboveGround>1</bldg:storeysAboveGround>
      <bldg:storeyHeightsAboveGround uom="#m">3.0</bldg:storeyHeightsAboveGround>
      <bldg:lod1Solid>
        <gml:Solid>
          <gml:exterior>
            <gml:CompositeSurface >
              <!-- Face Side 1 -->
              <gml:surfaceMember>
                <gml:Polygon>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:posList>458875.0 5438350.0 112.0 458885.0 5438350.0 112.0 458885.0 5438350.0 116.0 458875.0
                        5438350.0 116.0 458875.0 5438350.0 112.0 </gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
              <!-- Face Side 2 -->
```

```
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList>458885.0 5438350.0 112.0 458885.0 5438355.0 112.0 458885.0 5438355.0 116.0 458885.0
                      5438350.0 116.0 458885.0 5438350.0 112.0 </gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
            <!-- Face Side 3 -->
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList>458885.0 5438355.0 112.0 458875.0 5438355.0 112.0 458875.0 5438355.0 116.0 458885.0
                      5438355.0 116.0 458885.0 5438355.0 112.0 </gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
            <!-- Face Side 4 -->
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList>458875.0 5438355.0 112.0 458875.0 5438350.0 112.0 458875.0 5438350.0 116.0 458875.0
                      5438355.0 116.0 458875.0 5438355.0 112.0 </gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
            <!-- Face Top -->
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList>458875.0 5438350.0 116.0 458885.0 5438350.0 116.0 458885.0 5438355.0 116.0 458875.0
                      5438355.0 116.0 458875.0 5438350.0 116.0 </gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
            <!-- Face Bottom -->
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList>458875.0 5438350.0 112.0 458875.0 5438355.0 112.0 458885.0 5438355.0 112.0 458885.0
                      5438350.0 112.0 458875.0 5438350.0 112.0 </gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
          </gml:CompositeSurface>
        </gml:exterior>
      </gml:Solid>
    </bldg:lod1Solid>
    <bldg:address>
      <Address>
        <xalAddress>
          <xAL:AddressDetails>
            <xAL:Country>
              <xAL:CountryName>Germany</xAL:CountryName>
              <xAL:Locality Type="Town">
                <xAL:LocalityName>Eggenstein-Leopoldshafen</xAL:LocalityName>
                <xAL:Thoroughfare Type="Street">
                  <xAL:ThoroughfareNumber>1</xAL:ThoroughfareNumber>
                  <xAL:ThoroughfareName>Hermann-von-Helmholtz-Platz</xAL:ThoroughfareName>
                </xAL:Thoroughfare>
                <xAL:PostalCode>
                  <xAL:PostalCodeNumber>76344</xAL:PostalCodeNumber>
                </xAL:PostalCode>
              </xAL:Locality>
            </xAL:Country>
          </xAL:AddressDetails>
```

```
            </xalAddress>
            <multiPoint>
              <gml:MultiPoint>
                <gml:pointMember>
                  <gml:Point>
                    <gml:pos srsDimension="3">458880.0 5438352.6 112.0 </gml:pos>
                  </gml:Point>
                </gml:pointMember>
              </gml:MultiPoint>
            </multiPoint>
          </Address>
        </bldg:address>
      </bldg:Building>
    </cityObjectMember>
    <cityObjectMember>
      <dem:ReliefFeature gml:id="GML_6bb30328-7599-4500-90ef-766fde6aa67b">
        <gml:name>Example TIN LOD1</gml:name>
        <dem:lod>1</dem:lod>
        <dem:reliefComponent>
          <dem:TINRelief gml:id="GML_4eb161b0-aa7e-4087-937c-5c4c427c7fc9">
            <gml:name>Ground</gml:name>
            <dem:lod>1</dem:lod>
            <dem:tin>
              <gml:TriangulatedSurface>
                <gml:trianglePatches>
                  <gml:Triangle>
                    <gml:exterior>
                      <gml:LinearRing>
                        <gml:posList>458868.0 5438362.0 112.0 458875.0 5438355.0 112.0 458883.0 5438362.0 114.0 458868.0
                          5438362.0 112.0 </gml:posList>
                      </gml:LinearRing>
                    </gml:exterior>
                  </gml:Triangle>
                  <gml:Triangle>
                    …
                  </gml:Triangle>
                    ... (more triangles)
                </gml:trianglePatches>
              </gml:TriangulatedSurface>
            </dem:tin>
          </dem:TINRelief>
        </dem:reliefComponent>
      </dem:ReliefFeature>
    </cityObjectMember>
</CityModel>
```

Listing 4: Excerpt from the CityGML dataset for a building in LOD1 visualised in Fig. 76.

G.3 Example of a CityGML dataset for a building in LOD2

The building model from the previous chapters is now represented in LOD2. The model reflects the actual roof structure and contains boundary surfaces (*bldg:boundedBy*) which semantically classify the surfaces of the exterior building shell (*bldg:RoofSurface*, *bldg:WallSurface*, and *bldg:GroundSurface*). In addition to the thematic boudary surfaces, the building geometry is also described using an LOD2 solid geometry (*gml:Solid*). According to conformance requirement no. 4 of the *Building* module (cf. chapter 10.3.9), this solid geometry must reference the geometry of the boundary surfaces by using the GML3 XLink mechanism (*xlink:href*, printed in bold in the following Listing 5).

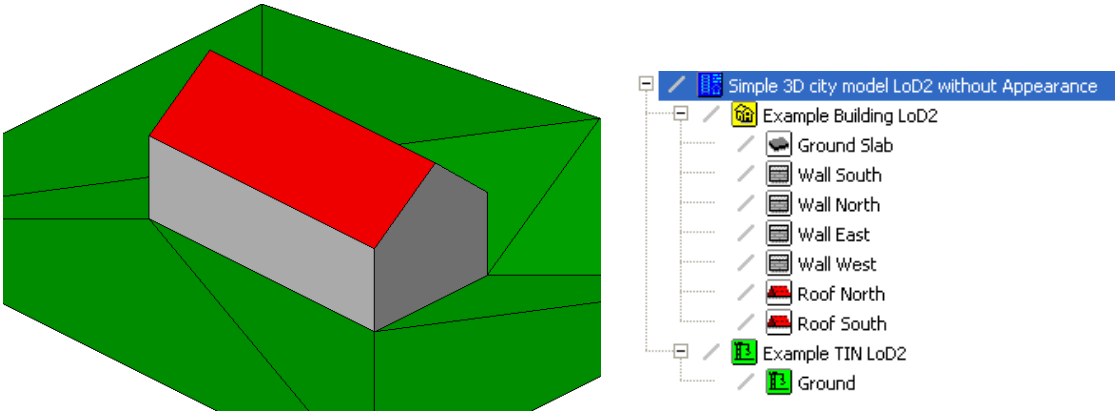

Fig. 77: Example of a CityGML building model in LOD2 (left: 3D graphic; right: model hierachy).

```
<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.opengis.net/citygml/2.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
  xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
  xsi:schemaLocation="http://www.opengis.net/citygml/building/2.0 http://schemas.opengis.net/citygml/building/2.0/building.xsd
                      http://www.opengis.net/citygml/relief/2.0 http://schemas.opengis.net/citygml/relief/2.0/relief.xsd ">
  <gml:name>Simple 3D city model LOD2 without Appearance</gml:name>
  <gml:boundedBy>
    <gml:Envelope srsDimension="3" srsName="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783">
      <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
      <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <cityObjectMember>
    <bldg:Building gml:id="GML_7b1a5a6f-ddad-4c3d-a507-3eb9ee0a8e68">
      <gml:name>Example Building LOD2 </gml:name>
      … (further attributes see LOD1 example)
      <bldg:lod2Solid>
        <gml:Solid>
          <gml:exterior>
            <gml:CompositeSurface>
              <!-- Ground Slab -->
              <gml:surfaceMember xlink:href="#GML_d3981803-d4b0-4b5b-969c-53f657594757" />
              <!-- Wall South -->
              <gml:surfaceMember xlink:href="#GML_1d350a50-6acc-4d3c-8c28-326ca4305fd1" />
              <!-- Wall North -->
              <gml:surfaceMember xlink:href="#GML_d3909000-2f18-4472-8886-1c127ea67df1" />
              <!-- Wall East -->
              <gml:surfaceMember xlink:href="#GML_6286ffa9-3811-4796-a92f-3fd037c8e668" />
              <!-- Wall West -->
              <gml:surfaceMember xlink:href="#GML_5cc4fd92-d5de-4dd8-971e-892c91da2d9f" />
              <!-- Roof North -->
              <gml:surfaceMember xlink:href="#GML_ec6a8966-58d9-4894-8edd-9aceb91b923f" />
              <!-- Roof South -->
              <gml:surfaceMember xlink:href="#GML_b41dc792-5da6-4cd9-8f85-247583f305e3" />
            </gml:CompositeSurface>
```

```
      </gml:exterior>
    </gml:Solid>
  </bldg:lod2Solid>
  <bldg:boundedBy>
    <bldg:GroundSurface>
      <gml:name>Ground Slab</gml:name>
      <bldg:lod2MultiSurface>
        <gml:MultiSurface>
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_d3981803-d4b0-4b5b-969c-53f657594757">
              <gml:exterior>
                <gml:LinearRing>
                  <gml:posList>458875.0 5438350.0 112.0 458875.0 5438355.0 112.0 458885.0 5438355.0 112.0 458885.0 5438350.0
                  112.0 458875.0 5438350.0 112.0 </gml:posList>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod2MultiSurface>
    </bldg:GroundSurface>
  </bldg:boundedBy>
  <bldg:boundedBy>
    <bldg:WallSurface>
      <gml:name>Wall South</gml:name>
      <bldg:lod2MultiSurface>
        <gml:MultiSurface>
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_1d350a50-6acc-4d3c-8c28-326ca4305fd1">
              <gml:exterior>
                <gml:LinearRing>
                  <gml:posList>458875.0 5438350.0 112.0 458885.0 5438350.0 112.0 458885.0 5438350.0 115.0 458875.0 5438350.0
                  115.0 458875.0 5438350.0 112.0 </gml:posList>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod2MultiSurface>
    </bldg:WallSurface>
  </bldg:boundedBy>
  <bldg:boundedBy>
    <bldg:WallSurface>
      <gml:name>Wall North</gml:name>
      <bldg:lod2MultiSurface>
        <gml:MultiSurface>
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_d3909000-2f18-4472-8886-1c127ea67df1">
              …
            </gml:Polygon>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod2MultiSurface>
    </bldg:WallSurface>
  </bldg:boundedBy>
  <bldg:boundedBy>
    <bldg:WallSurface>
      <gml:name>Wall East</gml:name>
      <bldg:lod2MultiSurface>
        <gml:MultiSurface>
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_6286ffa9-3811-4796-a92f-3fd037c8e668">
              …
            </gml:Polygon>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod2MultiSurface>
    </bldg:WallSurface>
  </bldg:boundedBy>
  <bldg:boundedBy>
    <bldg:WallSurface>
      <gml:name>Wall West</gml:name>
      <bldg:lod2MultiSurface>
        <gml:MultiSurface>
          <gml:surfaceMember>
```

```
                <gml:Polygon gml:id="GML_5cc4fd92-d5de-4dd8-971e-892c91da2d9f">
                  …
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:MultiSurface>
          </bldg:lod2MultiSurface>
        </bldg:WallSurface>
      </bldg:boundedBy>
      <bldg:boundedBy>
        <bldg:RoofSurface>
          <gml:name>Roof North</gml:name>
          <bldg:lod2MultiSurface>
            <gml:MultiSurface>
              <gml:surfaceMember>
                <gml:Polygon gml:id="GML_ec6a8966-58d9-4894-8edd-9aceb91b923f">
                  …
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:MultiSurface>
          </bldg:lod2MultiSurface>
        </bldg:RoofSurface>
      </bldg:boundedBy>
      <bldg:boundedBy>
        <bldg:RoofSurface>
          <gml:name>Roof South</gml:name>
          <bldg:lod2MultiSurface>
            <gml:MultiSurface>
              <gml:surfaceMember>
                <gml:Polygon gml:id="GML_b41dc792-5da6-4cd9-8f85-247583f305e3">
                  …
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:MultiSurface>
          </bldg:lod2MultiSurface>
        </bldg:RoofSurface>
      </bldg:boundedBy>
      <bldg:address>
        <Address>
          <xalAddress>
            <xAL:AddressDetails>
              …
            </xAL:AddressDetails>
          </xalAddress>
          <multiPoint>
            <gml:MultiPoint>
              <gml:pointMember>
                <gml:Point>
                  <gml:pos srsDimension="3">458880.0 5438352.7 112.0 </gml:pos>
                </gml:Point>
              </gml:pointMember>
            </gml:MultiPoint>
          </multiPoint>
        </Address>
      </bldg:address>
    </bldg:Building>
  </cityObjectMember>
  <cityObjectMember>
    <dem:ReliefFeature gml:id="GML_6bb30328-7599-4500-90ef-766fde6aa67b">
      <gml:name>Example TIN LOD2</gml:name>
      <dem:lod>2</dem:lod>
      <dem:reliefComponent>
        <dem:TINRelief gml:id="GML_4eb161b0-aa7e-4087-937c-5c4c427c7fc9">
          <gml:name>Ground</gml:name>
          <dem:lod>2</dem:lod>
          <dem:tin>
            …
          </dem:tin>
        </dem:TINRelief>
      </dem:reliefComponent>
    </dem:ReliefFeature>
  </cityObjectMember>
</CityModel>
```

Listing 5: Excerpt from the CityGML dataset for a building in LOD2 visualised in Fig. 77.

G.4 Example of a CityGML dataset for a building in LOD2 with an adjacent building part illustrating CityGML's topology representation

This example illustrates CityGML's topology representation which uses the XLink mechanism of GML3 (cf. chapter 8.1). The LOD2 model of annex G.3 is extended by placing an adjacent garage next to the building (cf. Fig. 78). The garage is modelled as building part (*bldg:BuildingPart*) and shares a common surface geometry with the building shell. For both the building and the garage thematic boundary surfaces (*bldg:boundedBy*) as well as separate solid geometries are given. The wall surface of the building where the garage touches the building is depicted in Fig. 79. For the building, the surface geometry is split into a non-shared and a shared part (see *bldg:WallSurface* with *gml:name* "Wall East"). The latter is referenced by the garage both by its corresponding *bldg:WallSurface* (*gml:name* "Garage Wall West") and its solid geometry. Since the orientation of the shared surface has to be reversed in the context of the garage, the XLink reference is embraced by a *gml:OrientableSurface* element. The XLinks pointing from the garage to the building geometry explicitly denote the topological adjacency relation between both features.

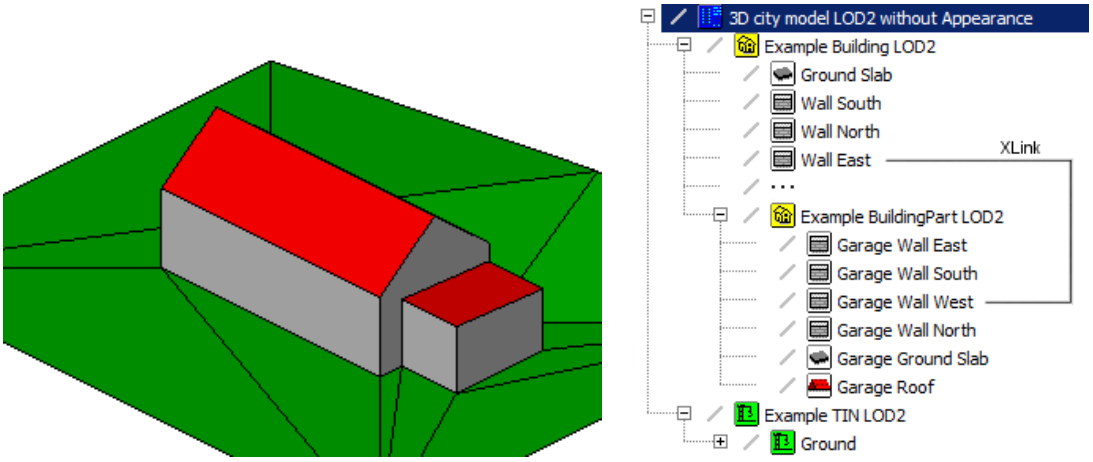

Fig. 78: Example of a CityGML building model in LOD2 with an adjacent building part (garage) which shares a common boundary surface (left: 3D graphic; right: model hierachy).

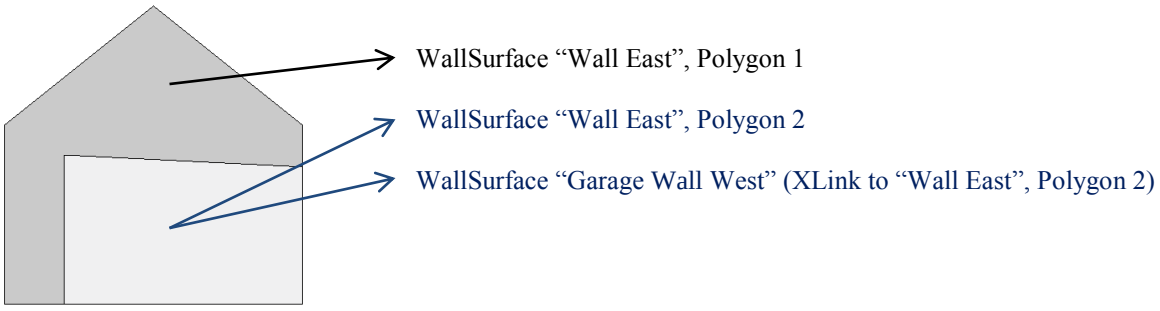

Fig. 79: The boundary surface where the garage touches the building. For the building, the geometry is split into a non-shared part ("Wall East", Polygon 1) and a shared part ("Wall East", Polygon 2). Only the latter is referenced by the garage using the XLink mechanism.

```
<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.opengis.net/citygml/2.0"
  xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0" xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:gml="http://www.opengis.net/gml" xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  schemaLocation="http://www.opengis.net/citygml/building/2.0 http://schemas.opengis.net/citygml/building/2.0/building.xsd
                  http://www.opengis.net/citygml/relief/2.0 http://schemas.opengis.net/citygml/relief/2.0/relief.xsd">
  <gml:name>3D city model LOD2 without Appearance</gml:name>
  <gml:boundedBy>
    <gml:Envelope srsDimension="3" srsName="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783">
      <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
      <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <cityObjectMember>
```

```
<bldg:Building gml:id="GML_7b1a5a6f-ddad-4c3d-a507-3eb9ee0a8e68">
  <gml:name>Example Building LOD2</gml:name>
  …  (further attributes see LOD1 example)
  <bldg:lod2Solid>
    <gml:Solid>
      <gml:exterior>
        <gml:CompositeSurface>
          <!-- Ground Slab -->
          <gml:surfaceMember xlink:href="#GML_d3981803-d4b0-4b5b-969c-53f657594757"/>
          <!-- Wall South -->
          <gml:surfaceMember xlink:href="#GML_1d350a50-6acc-4d3c-8c28-326ca4305fd1"/>
          <!-- Wall North -->
          <gml:surfaceMember xlink:href="#GML_d3909000-2f18-4472-8886-1c127ea67df1"/>
          <!-- Wall East 1 -->
          <gml:surfaceMember xlink:href="#GML_56d1dd88-36dd-4d1e-bff0-3305fbffa778"/>
          <!-- Wall East 2 -->
          <gml:surfaceMember xlink:href="#GML_9f0465e6-f316-4f89-a9bd-eb21934ffe84"/>
          <!-- Wall West -->
          <gml:surfaceMember xlink:href="#GML_5cc4fd92-d5de-4dd8-971e-892c91da2d9f"/>
          <!-- Roof North -->
          <gml:surfaceMember xlink:href="#GML_ec6a8966-58d9-4894-8edd-9aceb91b923f"/>
          <!-- Roof South -->
          <gml:surfaceMember xlink:href="#GML_b41dc792-5da6-4cd9-8f85-247583f305e3"/>
        </gml:CompositeSurface>
      </gml:exterior>
    </gml:Solid>
  </bldg:lod2Solid>
  <bldg:boundedBy>
    <bldg:GroundSurface>
      <gml:name>Ground Slab</gml:name>
      <bldg:lod2MultiSurface>
        <gml:MultiSurface>
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_d3981803-d4b0-4b5b-969c-53f657594757">
              <gml:exterior>
                <gml:LinearRing>
                  <gml:posList>458875.0 5438350.0 112.0 458875.0 5438355.0 112.0 458885.0 5438355.0 112.0 458885.0 5438350.0
                    112.0 458875.0 5438350.0 112.0 </gml:posList>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod2MultiSurface>
    </bldg:GroundSurface>
  </bldg:boundedBy>
  …
  <bldg:boundedBy>
    <bldg:WallSurface>
      <gml:name>Wall East</gml:name>
      <bldg:lod2MultiSurface>
        <gml:MultiSurface>
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_56d1dd88-36dd-4d1e-bff0-3305fbffa778">
              <gml:exterior>
                <gml:LinearRing>
                  <gml:posList>458885.0 5438350.0 112.0 458885.0 5438351.0 112.0 458885.0 5438351.0 114.5 458885.0 5438355.0
                    114.3 458885.0 5438355.0 115.0 458885.0 5438352.5 117.0 458885.0 5438350.0 115.0 458885.0 5438350.0
                    112.0</gml:posList>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_9f0465e6-f316-4f89-a9bd-eb21934ffe84">
              <gml:exterior>
                <gml:LinearRing>
                  <gml:posList>458885.0 5438355.0 112.0 458885.0 5438355.0 114.3 458885.0 5438351.0 114.5 458885.0 5438351.0
                    112.0 458885.0 5438355.0 112.0</gml:posList>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod2MultiSurface>
    </bldg:WallSurface>
```

```
</bldg:boundedBy>
…
<bldg:consistsOfBuildingPart>
  <bldg:BuildingPart gml:id="GMLID_BUI379228_1244_301">
    <gml:name>Example BuildingPart LOD2</gml:name>
    <bldg:function
        codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_function.xml">1630</bldg:function>
    <bldg:yearOfConstruction>1985</bldg:yearOfConstruction>
    <bldg:roofType
        codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml">1010</bldg:roofType>
    <bldg:measuredHeight uom="#m">2.5</bldg:measuredHeight>
    <bldg:lod2Solid>
      <gml:Solid>
        <gml:exterior>
          <gml:CompositeSurface>
            <!-- Garage Ground Slab -->
            <gml:surfaceMember xlink:href="#GML_2e1ff653-b62b-41ee-9f99-d6852ae7d567"/>
            <!-- Garage Wall South -->
            <gml:surfaceMember xlink:href="#GML_f3f56c7b-7e59-47bc-ba03-d841032f1a37"/>
            <!-- Garage Wall North -->
            <gml:surfaceMember xlink:href="#GML_5339468c-b2cb-4a99-9eb5-8b0660fb26d3"/>
            <!-- Garage Wall East -->
            <gml:surfaceMember xlink:href="#GML_dab75f49-f6f8-4490-b86b-450b613e1fc2"/>
            <!-- Garage Wall West (identical with Wall East 2 of Building) -->
            <gml:surfaceMember>
              <gml:OrientableSurface orientation="-">
                <gml:baseSurface xlink:href="#GML_9f0465e6-f316-4f89-a9bd-eb21934ffe84"/>
              </gml:OrientableSurface>
            </gml:surfaceMember>
            <!-- Garage Roof -->
            <gml:surfaceMember xlink:href="#GML_7996bef1-f045-4704-be27-db27430d4f70"/>
          </gml:CompositeSurface>
        </gml:exterior>
      </gml:Solid>
    </bldg:lod2Solid>
    <bldg:boundedBy>
      <bldg:WallSurface>
        <gml:name>Garage Wall East</gml:name>
        <bldg:lod2MultiSurface>
          <gml:MultiSurface>
            <gml:surfaceMember>
              <gml:Polygon gml:id="GML_dab75f49-f6f8-4490-b86b-450b613e1fc2">
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList>458887.5 5438355.0 114.3 458887.5 5438351.0 114.5 458887.5 5438351.0 112.0 458887.5
                      5438355.0 112.0 458887.5 5438355.0 114.3</gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
          </gml:MultiSurface>
        </bldg:lod2MultiSurface>
      </bldg:WallSurface>
    </bldg:boundedBy>
    …
    <bldg:boundedBy>
      <bldg:WallSurface>
        <gml:name>Garage Wall West</gml:name>
        <bldg:lod2MultiSurface>
          <gml:MultiSurface>
            <gml:surfaceMember>
              <!-- identical with Wall East 2 of Building -->
              <gml:OrientableSurface orientation="-">
                <gml:baseSurface xlink:href="#GML_9f0465e6-f316-4f89-a9bd-eb21934ffe84"/>
              </gml:OrientableSurface>
            </gml:surfaceMember>
          </gml:MultiSurface>
        </bldg:lod2MultiSurface>
      </bldg:WallSurface>
    </bldg:boundedBy>
    …
  </bldg:BuildingPart>
</bldg:consistsOfBuildingPart>
<bldg:address>
  <Address>
    <xalAddress>
```

```
            <xAL:AddressDetails>
              …
            </xAL:AddressDetails>
          </xalAddress>
          <multiPoint>
            …
          </multiPoint>
        </Address>
      </bldg:address>
    </bldg:Building>
  </cityObjectMember>
  <cityObjectMember>
    <dem:ReliefFeature gml:id="GML_6bb30328-7599-4500-90ef-766fde6aa67b">
      <gml:name>Example TIN LOD2</gml:name>
      <dem:lod>2</dem:lod>
      <dem:reliefComponent>
        <dem:TINRelief gml:id="GML_4eb161b0-aa7e-4087-937c-5c4c427c7fc9">
          <gml:name>Ground</gml:name>
          <dem:lod>2</dem:lod>
          <dem:tin>
            …
          </dem:tin>
        </dem:TINRelief>
      </dem:reliefComponent>
    </dem:ReliefFeature>
  </cityObjectMember>
</CityModel>
```

Listing 6: Excerpt from the CityGML dataset for a building in LOD2 having an adjacent garage as visualised in Fig. 78.

G.5 Example of a CityGML dataset for a building in LOD3

The LOD3 building model illustrated in this chapter (cf. Fig. 80) adds doors (*bldg:Door*), windows (*bldg:Window*) and roof overhangs (*bldg:RoofSurface*) to the LOD2 representation of the previous chapters. Again, a solid geometry for the LOD3 of the building is realized by referencing the geometries of the thematic boundary surfaces using the GML3 XLink mechanism (*xlink:href*). In order to get a valid solid geometry, the roof surfaces are geometrically split into the roof slab and the roof overhang parts. Only the geometry of the roof slab is referenced by the solid. Walls composed of serveral surfaces (e.g. reveals) are modeled as *gml:CompositeSurface* which then is referenced by the building solid. Boundary surfaces containing openings for doors or windows are modeled with polygons having one exterior and serval interior linear rings (according to conformance requirement no. 8 of the *Building* module, cf. chapter 10.4.8).

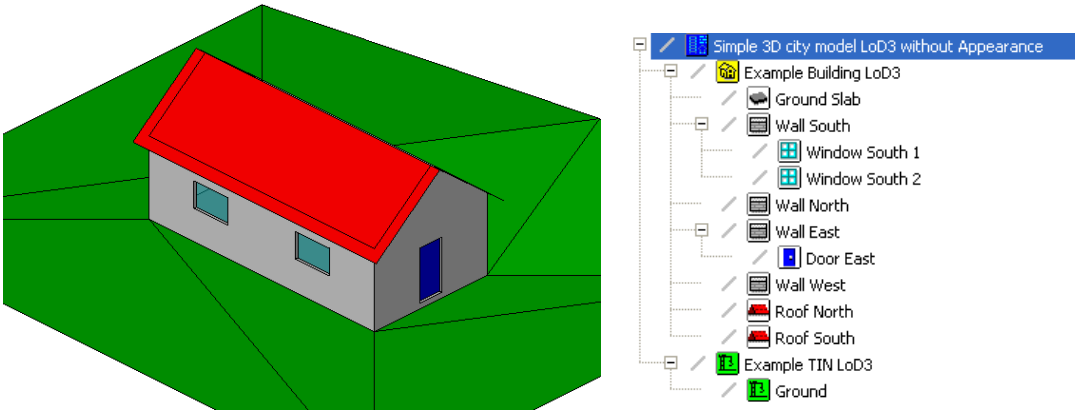

Fig. 80: Example of a CityGML building model in LOD3 (left: 3D graphic; right: model hierachy).

```
<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.opengis.net/citygml/2.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
  xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
  xsi:schemaLocation="http://www.opengis.net/citygml/building/2.0 http://schemas.opengis.net/citygml/building/2.0/building.xsd
                      http://www.opengis.net/citygml/relief/2.0 http://schemas.opengis.net/citygml/relief/2.0/relief.xsd ">
  <gml:name>Simple 3D city model LOD3 without Appearance</gml:name>
  <gml:boundedBy>
    <gml:Envelope srsDimension="3" srsName="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783">
      <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
      <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <cityObjectMember>
    <bldg:Building gml:id="GML_7b1a5a6f-ddad-4c3d-a507-3eb9ee0a8e68">
      <gml:name>Example Building LOD3 </gml:name>
        … (further attributes see LOD1 example)
      <bldg:boundedBy>
        <bldg:GroundSurface>
          <gml:name>Ground Slab</gml:name>
            … (see LOD2 example)
        </bldg:GroundSurface>
      </bldg:boundedBy>
      <bldg:boundedBy>
        <bldg:WallSurface>
          <gml:name>Wall South</gml:name>
          <bldg:lod3MultiSurface>
            <gml:MultiSurface>
              <gml:surfaceMember>
                <gml:CompositeSurface gml:id="GML_1d350a50-6acc-4d3c-8c28-326ca4305fd1">
                  <gml:surfaceMember>
                    <gml:Polygon gml:id="PolyID10204_1916_571790_369478">
                      …
```

```
            </gml:Polygon>
          </gml:surfaceMember>
          <gml:surfaceMember>
            <gml:Polygon gml:id="PolyID10205_105_876837_53833">
              <gml:exterior>
                <gml:LinearRing>
                  <gml:posList>458875.0 5438350.0 112.0 458885.0 5438350.0 112.0 458885.0 5438350.0 115.0
                    458875.0 5438350.0 115.0 458875.0 5438350.0 112.0 </gml:posList>
                </gml:LinearRing>
              </gml:exterior>
              <gml:interior>
                <gml:LinearRing>
                  <gml:posList>458877.0 5438350.0 114.2 458878.5 5438350.0 114.2 458878.5 5438350.0 113.2
                    458877.0 5438350.0 113.2 458877.0 5438350.0 114.2 </gml:posList>
                </gml:LinearRing>
              </gml:interior>
              <gml:interior>
                <gml:LinearRing>
                  <gml:posList>458881.5 5438350.0 114.2 458883.0 5438350.0 114.2 458883.0 5438350.0 113.2
                    458881.5 5438350.0 113.2 458881.5 5438350.0 114.2 </gml:posList>
                </gml:LinearRing>
              </gml:interior>
            </gml:Polygon>
          </gml:surfaceMember>
          <gml:surfaceMember>
            … (more surface members of the WallSurface)
          </gml:surfaceMember>
        </gml:CompositeSurface>
      </gml:surfaceMember>
    </gml:MultiSurface>
  </bldg:lod3MultiSurface>
  <bldg:opening>
    <bldg:Window gml:id="GML_3b09d6a5-4c24-4847-a8a2-e97475e3de47">
      <gml:name>Window South 1</gml:name>
      <bldg:lod3MultiSurface>
        <gml:MultiSurface>
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_5e07e2cc-c28c-480e-880f-dfdfe287bb9e">
            …
            </gml:Polygon>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod3MultiSurface>
    </bldg:Window>
  </bldg:opening>
  <bldg:opening>
    <bldg:Window gml:id="GML_f75f01cc-c584-4a62-b34a-4a0e2640550d">
      <gml:name>Window South 2</gml:name>
      <bldg:lod3MultiSurface>
        <gml:MultiSurface>
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_d0ea2b6b-7992-4284-9a20-957a6c5c1cea">
            …
            </gml:Polygon>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod3MultiSurface>
    </bldg:Window>
  </bldg:opening>
</bldg:WallSurface>
</bldg:boundedBy>
<bldg:boundedBy>
  <bldg:WallSurface>
    <gml:name>Wall North</gml:name>
    … (see LOD2 example)
  </bldg:WallSurface>
</bldg:boundedBy>
<bldg:boundedBy>
  <bldg:WallSurface>
    <gml:name>Wall East</gml:name>
    <bldg:lod3MultiSurface>
      <gml:MultiSurface>
        <gml:surfaceMember>
          <gml:CompositeSurface gml:id="GML_6286ffa9-3811-4796-a92f-3fd037c8e668">
          ...
```

```
            </gml:CompositeSurface>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod3MultiSurface>
      <bldg:opening>
        <bldg:Door gml:id="GML_93096bbb-5155-47fb-ae2c-e2f9327f3007">
          <gml:name>Door East</gml:name>
          <bldg:lod3MultiSurface>
            <gml:MultiSurface>
              <gml:surfaceMember>
                <gml:Polygon gml:id="GML_8f988da9-22d7-41e5-ae94-880afd46a3c9">
                …
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:MultiSurface>
          </bldg:lod3MultiSurface>
        </bldg:Door>
      </bldg:opening>
    </bldg:WallSurface>
  </bldg:boundedBy>
  <bldg:boundedBy>
    <bldg:WallSurface>
      <gml:name>Wall West</gml:name>
      … (see LOD2 example)
    </bldg:WallSurface>
  </bldg:boundedBy>
  <bldg:boundedBy>
    <bldg:RoofSurface>
      <gml:name>Roof North</gml:name>
      <bldg:lod3MultiSurface>
        <gml:MultiSurface>
          <!-- Roof slab -->
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_ec6a8966-58d9-4894-8edd-9aceb91b923f">
            … (see LOD2 example)
            </gml:Polygon>
          </gml:surfaceMember>
          <!-- Roof overhanging -->
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_70fa738e-80a4-4774-8a3b-322f037fa482">
              <gml:exterior>
                <gml:LinearRing>
                  <gml:posList>458874.6 5438352.5 117 458875 5438352.5 117 458875 5438355 115 458885 5438355 115 458885
                  5438352.5 117 458885.4 5438352.5 117 458885.4 5438355.312347524 114.75012198097823 458874.6
                  5438355.312347524 114.75012198097823 458874.6 5438352.5 117 </gml:posList>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod3MultiSurface>
    </bldg:RoofSurface>
  </bldg:boundedBy>
  <bldg:boundedBy>
    <bldg:RoofSurface>
      <gml:name>Roof South</gml:name>
      <bldg:lod3MultiSurface>
        <!-- Roof slab -->
        <gml:MultiSurface>
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_b41dc792-5da6-4cd9-8f85-247583f305e3">
            … (see LOD2 example)
            </gml:Polygon>
          </gml:surfaceMember>
          <!-- Roof overhanging -->
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_db6d8edc-4870-4523-a606-d440f36f8ec8">
            …
            </gml:Polygon>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod3MultiSurface>
    </bldg:RoofSurface>
  </bldg:boundedBy>
  <bldg:lod3Solid>
    <gml:Solid>
```

```
          <gml:exterior>
            <gml:CompositeSurface>
              <!-- Ground Slab -->
              <gml:surfaceMember xlink:href="#GML_d3981803-d4b0-4b5b-969c-53f657594757"/>
              <!-- Wall South -->
              <gml:surfaceMember xlink:href="#GML_1d350a50-6acc-4d3c-8c28-326ca4305fd1"/>
              <!-- Window South 1 -->
              <gml:surfaceMember xlink:href="#GML_5e07e2cc-c28c-480e-880f-dfdfe287bb9e"/>
              <!-- Window South 2 -->
              <gml:surfaceMember xlink:href="#GML_d0ea2b6b-7992-4284-9a20-957a6c5c1cea"/>
              <!-- Wall North -->
              <gml:surfaceMember xlink:href="#GML_d3909000-2f18-4472-8886-1c127ea67df1"/>
              <!-- Wall East -->
              <gml:surfaceMember xlink:href="#GML_6286ffa9-3811-4796-a92f-3fd037c8e668"/>
              <!-- Door East -->
              <gml:surfaceMember xlink:href="#GML_8f988da9-22d7-41e5-ae94-880afd46a3c9"/>
              <!-- Wall West -->
              <gml:surfaceMember xlink:href="#GML_5cc4fd92-d5de-4dd8-971e-892c91da2d9f"/>
              <!-- Roof Slab North -->
              <gml:surfaceMember xlink:href="#GML_ec6a8966-58d9-4894-8edd-9aceb91b923f"/>
              <!-- Roof Slab South -->
              <gml:surfaceMember xlink:href="#GML_b41dc792-5da6-4cd9-8f85-247583f305e3"/>
            </gml:CompositeSurface>
          </gml:exterior>
        </gml:Solid>
      </bldg:lod3Solid>
      <bldg:address>
        <Address>
        ...  (see LOD1 example)
        </Address>
      </bldg:address>
    </bldg:Building>
  </cityObjectMember>
  <cityObjectMember>
    <dem:ReliefFeature gml:id="GML_6bb30328-7599-4500-90ef-766fde6aa67b">
          …  (see LOD1 example)
    </dem:ReliefFeature>
  </cityObjectMember>
</CityModel>
```

Listing 7: Excerpt from the CityGML dataset for the buildings in LOD3 visualised in Fig. 80.

G.6 Example of a CityGML dataset for a building in LOD4

In LOD4, the building is completed with the representation of the building interior (cf. Fig. 81). The model contains a room (*bldg:Room*), which is equipped with a rocking chair (*bldg:BuildingFurniture*). The room is bounded by interior boundary surfaces (*bldg:InteriorWallSurface*, *bldg:FloorSurface*, *bldg:CeilingSurface*, associated through the *bldg:boundedBy* property of the room) whose geometries are referenced by the LOD4 solid geometry of the room (*xlink:href*). If the normal vector of an interior boundary surface is pointing into the room, its orientation has to be flipped using an orientable surface (*gml:OrientableSurface*) when referenced from the solid in order to create a valid solid geometry (for a *gml:Solid*, the normal vectors of the surfaces bounding the volume have to point outwards the volume).

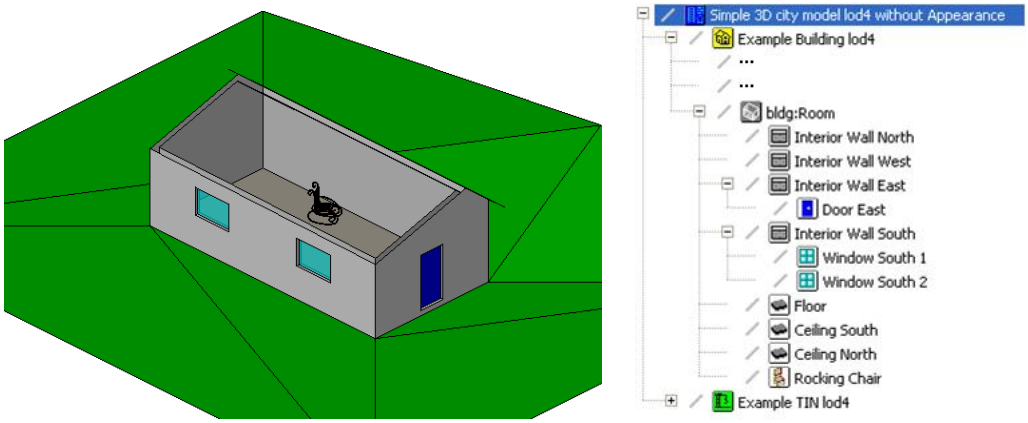

Fig. 81: Example of a CityGML building model in LOD4. The roof surfaces are not shown in order to visualize the interior boundary surfaces as well as the building furniture (left: 3D graphic; right: model hierachy).

```
<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.opengis.net/citygml/2.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
  xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
  xsi:schemaLocation="http://www.opengis.net/citygml/building/2.0 http://schemas.opengis.net/citygml/building/2.0/building.xsd
                      http://www.opengis.net/citygml/relief/2.0 http://schemas.opengis.net/citygml/relief/2.0/relief.xsd ">
  <gml:name>Simple 3D city model LOD4 without Appearance</gml:name>
  <gml:boundedBy>
    <gml:Envelope srsDimension="3" srsName="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783">
      <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
      <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <cityObjectMember>
    <bldg:Building gml:id="GML_7b1a5a6f-ddad-4c3d-a507-3eb9ee0a8e68">
      <gml:name>Example Building LOD4 </gml:name>
        … (further attributes see LOD1 example)
      <bldg:boundedBy>
        … (outer shell see LOD3 example)
      <bldg:lod4Solid>
        … (building solid representation see LOD3 example)
      </bldg:lod4Solid>
      <bldg:interiorRoom>
        <bldg:Room>
          <bldg:lod4Solid>
            <gml:Solid>
              <gml:exterior>
                <gml:CompositeSurface>
                  <!-- Floor -->
                  <gml:surfaceMember>
                    <gml:OrientableSurface orientation="-">
                      <gml:baseSurface xlink:href="#GML_fa89e511-39b2-46de-9a13-9f4621576a46"/>
                    </gml:OrientableSurface>
                  </gml:surfaceMember>
                  <!-- Interior Wall North -->
```

```
            <gml:surfaceMember>
              <gml:OrientableSurface orientation="-">
                <gml:baseSurface xlink:href="#GML_592ce9fa-0b98-4225-8d22-20eff4f86fc5"/>
              </gml:OrientableSurface>
            </gml:surfaceMember>
            <!-- Interior Wall West -->
            <gml:surfaceMember>
              <gml:OrientableSurface orientation="-">
                <gml:baseSurface xlink:href="#GML_a9fe597d-c338-43ad-a633-2a0beb273fac"/>
              </gml:OrientableSurface>
            </gml:surfaceMember>
            <!-- Interior Wall East -->
            <gml:surfaceMember>
              <gml:OrientableSurface orientation="-">
                <gml:baseSurface xlink:href="#GML_eaf1db16-56a3-4b86-ae19-2edbb604636f"/>
              </gml:OrientableSurface>
            </gml:surfaceMember>
            <!-- Door East -->
            <gml:surfaceMember>
              <gml:OrientableSurface orientation="+">
                <gml:baseSurface xlink:href="#GML_8f988da9-22d7-41e5-ae94-880afd46a3c9"/>
              </gml:OrientableSurface>
            </gml:surfaceMember>
            <!-- Interior Wall South -->
            <gml:surfaceMember>
              <gml:OrientableSurface orientation="-">
                <gml:baseSurface xlink:href="#GML_a718c157-c948-42cf-a786-0ce61044cff9"/>
              </gml:OrientableSurface>
            </gml:surfaceMember>
            <!-- Window South 1 -->
            <gml:surfaceMember>
              <gml:OrientableSurface orientation="+">
                <gml:baseSurface xlink:href="#GML_5e07e2cc-c28c-480e-880f-dfdfe287bb9e"/>
              </gml:OrientableSurface>
            </gml:surfaceMember>
            <!-- Window South 2 -->
            <gml:surfaceMember>
              <gml:OrientableSurface orientation="+">
                <gml:baseSurface xlink:href="#GML_d0ea2b6b-7992-4284-9a20-957a6c5c1cea"/>
              </gml:OrientableSurface>
            </gml:surfaceMember>
            <!-- Ceiling North -->
            <gml:surfaceMember>
              <gml:OrientableSurface orientation="-">
                <gml:baseSurface xlink:href="#GML_989aa5cf-ee07-4fd8-89b6-500a9d5ba8041"/>
              </gml:OrientableSurface>
            </gml:surfaceMember>
            <!-- Ceiling South -->
            <gml:surfaceMember>
              <gml:OrientableSurface orientation="-">
                <gml:baseSurface xlink:href="#GML_98841838-ee0b-402f-ba28-64ed61cb10f8"/>
              </gml:OrientableSurface>
            </gml:surfaceMember>
          </gml:CompositeSurface>
        </gml:exterior>
      </gml:Solid>
    </bldg:lod4Solid>
    <bldg:boundedBy>
      <bldg:InteriorWallSurface>
        <gml:name>Interior Wall North</gml:name>
        <bldg:lod4MultiSurface>
          <gml:MultiSurface>
            <gml:surfaceMember>
              <gml:Polygon gml:id="GML_592ce9fa-0b98-4225-8d22-20eff4f86fc5">
                …
              </gml:Polygon>
            </gml:surfaceMember>
          </gml:MultiSurface>
        </bldg:lod4MultiSurface>
      </bldg:InteriorWallSurface>
    </bldg:boundedBy>
    <bldg:boundedBy>
      <bldg:InteriorWallSurface>
        <gml:name>Interior Wall West</gml:name>
        <bldg:lod4MultiSurface>
          <gml:MultiSurface>
```

```
            <gml:surfaceMember>
              <gml:Polygon gml:id="GML_a9fe597d-c338-43ad-a633-2a0beb273fac">
                …
              </gml:Polygon>
            </gml:surfaceMember>
          </gml:MultiSurface>
        </bldg:lod4MultiSurface>
      </bldg:InteriorWallSurface>
    </bldg:boundedBy>
    <bldg:boundedBy>
      <bldg:InteriorWallSurface>
        <gml:name>Interior Wall East</gml:name>
        <bldg:lod4MultiSurface>
          <gml:MultiSurface>
            <gml:surfaceMember>
              <gml:CompositeSurface gml:id="GML_eaf1db16-56a3-4b86-ae19-2edbb604636f">
                <gml:surfaceMember>
                  …
                </gml:surfaceMember>
                <gml:surfaceMember>
                  …
                </gml:surfaceMember>
                <gml:surfaceMember>
                  …
                </gml:surfaceMember>
                <gml:surfaceMember>
                  …
                </gml:surfaceMember>
              </gml:CompositeSurface>
            </gml:surfaceMember>
          </gml:MultiSurface>
        </bldg:lod4MultiSurface>
        <bldg:opening>
          <bldg:Door>
            <gml:name>Door East</gml:name>
            <bldg:lod4MultiSurface>
              <gml:MultiSurface>
                <gml:surfaceMember>
                  <gml:OrientableSurface orientation="-">
                    <gml:baseSurface xlink:href="#GML_8f988da9-22d7-41e5-ae94-880afd46a3c9"> </gml:baseSurface>
                  </gml:OrientableSurface>
                </gml:surfaceMember>
              </gml:MultiSurface>
            </bldg:lod4MultiSurface>
          </bldg:Door>
        </bldg:opening>
      </bldg:InteriorWallSurface>
    </bldg:boundedBy>
    <bldg:boundedBy>
      <bldg:InteriorWallSurface>
        <gml:name>Interior Wall South</gml:name>
        <bldg:lod4MultiSurface>
          <gml:MultiSurface>
            <gml:surfaceMember>
              <gml:CompositeSurface gml:id="GML_a718c157-c948-42cf-a786-0ce61044cff9">
                <gml:surfaceMember>
                  …
                </gml:surfaceMember>
                <gml:surfaceMember>
                  …
                </gml:surfaceMember>
                <gml:surfaceMember>
                  …
                </gml:surfaceMember>
                <gml:surfaceMember>
                  …
                </gml:surfaceMember>
                <gml:surfaceMember>
                  …
                </gml:surfaceMember>
                <gml:surfaceMember>
                  …
                </gml:surfaceMember>
                <gml:surfaceMember>
                  <gml:Polygon gml:id="GML_cf0b79ba-f31f-4bae-a10f-5bcc85ce2cf6">
                    <gml:exterior>
```

```
                    …
                  </gml:exterior>
                  <gml:interior>
                    …
                  </gml:interior>
                  <gml:interior>
                    …
                  </gml:interior>
                </gml:Polygon>
              </gml:surfaceMember>
              <gml:surfaceMember>
                …
              </gml:surfaceMember>
              <gml:surfaceMember>
                …
              </gml:surfaceMember>
            </gml:CompositeSurface>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod4MultiSurface>
      <bldg:opening>
        <bldg:Window>
          <gml:name>Window South 1</gml:name>
          <bldg:lod4MultiSurface>
            <gml:MultiSurface>
              <gml:surfaceMember>
                <gml:OrientableSurface orientation="-">
                  <gml:baseSurface xlink:href="#GML_5e07e2cc-c28c-480e-880f-dfdfe287bb9e"> </gml:baseSurface>
                </gml:OrientableSurface>
              </gml:surfaceMember>
            </gml:MultiSurface>
          </bldg:lod4MultiSurface>
        </bldg:Window>
      </bldg:opening>
      <bldg:opening>
        <bldg:Window>
          <gml:name>Window South 2</gml:name>
          <bldg:lod4MultiSurface>
            <gml:MultiSurface>
              <gml:surfaceMember>
                <gml:OrientableSurface orientation="-">
                  <gml:baseSurface xlink:href="#GML_d0ea2b6b-7992-4284-9a20-957a6c5c1cea"> </gml:baseSurface>
                </gml:OrientableSurface>
              </gml:surfaceMember>
            </gml:MultiSurface>
          </bldg:lod4MultiSurface>
        </bldg:Window>
      </bldg:opening>
    </bldg:InteriorWallSurface>
  </bldg:boundedBy>
  <bldg:boundedBy>
    <bldg:FloorSurface>
      <gml:name>Floor</gml:name>
      <bldg:lod4MultiSurface>
        <gml:MultiSurface>
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_fa89e511-39b2-46de-9a13-9f4621576a46">
              …
            </gml:Polygon>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod4MultiSurface>
    </bldg:FloorSurface>
  </bldg:boundedBy>
  <bldg:boundedBy>
    <bldg:CeilingSurface>
      <gml:name>Ceiling South</gml:name>
      <bldg:lod4MultiSurface>
        <gml:MultiSurface>
          <gml:surfaceMember>
            <gml:Polygon gml:id="GML_989aa5cf-ee07-4fd8-89b6-500a9d5ba8041">
              …
            </gml:Polygon>
          </gml:surfaceMember>
        </gml:MultiSurface>
      </bldg:lod4MultiSurface>
```

```
            </bldg:CeilingSurface>
          </bldg:boundedBy>
          <bldg:boundedBy>
            <bldg:CeilingSurface>
              <gml:name>Ceiling North</gml:name>
              <bldg:lod4MultiSurface>
                <gml:MultiSurface>
                  <gml:surfaceMember>
                    <gml:Polygon gml:id="GML_98841838-ee0b-402f-ba28-64ed61cb10f8">
                      …
                    </gml:Polygon>
                  </gml:surfaceMember>
                </gml:MultiSurface>
              </bldg:lod4MultiSurface>
            </bldg:CeilingSurface>
          </bldg:boundedBy>
          <bldg:interiorFurniture>
            <bldg:BuildingFurniture>
              <gml:name>Rocking Chair</gml:name>
              <bldg:function codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/BuildingFurniture_function.xml"
              >1340</bldg:function>
              <bldg:lod4Geometry>
                <gml:MultiSurface>
                …
                </gml:MultiSurface>
              </bldg:lod4Geometry>
            </bldg:BuildingFurniture>
          </bldg:interiorFurniture>
        </bldg:Room>
      </bldg:interiorRoom>
      <bldg:address>
          …  (address see LOD1 example)
      </bldg:address>
    </bldg:Building>
  </cityObjectMember>
  <cityObjectMember>
    <dem:ReliefFeature gml:id="GML_6bb30328-7599-4500-90ef-766fde6aa67b">
      …  (see LOD1 example)
    </dem:ReliefFeature>
  </cityObjectMember>
</CityModel>
```

Listing 8: Excerpt from the CityGML dataset for the buildings in LOD4 visualised in Fig. 81.

G.7 Example of a CityGML dataset illustrating the appearance model

The following CityGML dataset is based on the simple building model from chapters G.2 and G.3 given in LOD1 and LOD2. Furthermore two separate appearance themes are defined – a summer theme and a winter theme – describing different visual appearances for the building and the surrounding terrain. Each LOD has an individual appearance for these specific themes.

Several concepts of CityGML's appearance model are used in this dataset. Regarding LOD1, an *X3DMaterial* object defines the material of the whole building which is applied to all of its surfaces. In addition, a *GeoreferencedTexture* is assigned both to the terrain and the roof surface of the building. In LOD2 the vertical surfaces of the building are texturised individually using *ParameterizedTexture* objects whereas the roof surfaces and the terrain again are described by a *GeoreferencedTexture*. The texture mapping for the *GeoreferencedTexture* objects is given inline (cf. Listing 9) as wells as using an ESRI world file (cf. Listing 11). The modelling approach results in four possible visualizations of the dataset that are represented in Fig. 82 and Fig. 83.

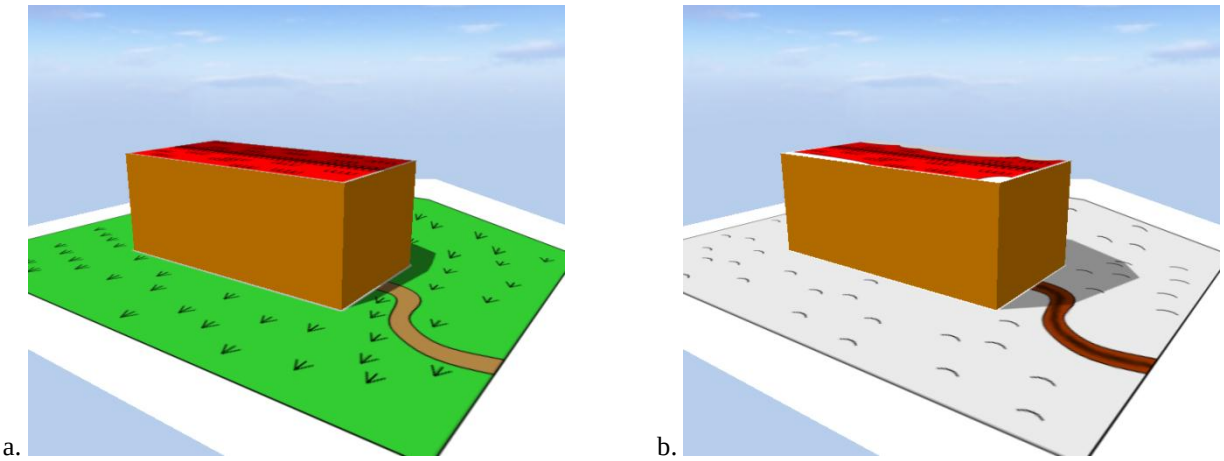

Fig. 82: Visualisation of a simple building in LOD1 using CityGML's appearance model. Two themes are defined for the building and the surrounding terrain: (a) theme showing the building in summer and (b) showing the building in winter (image: Hasso-Plattner-Institute).

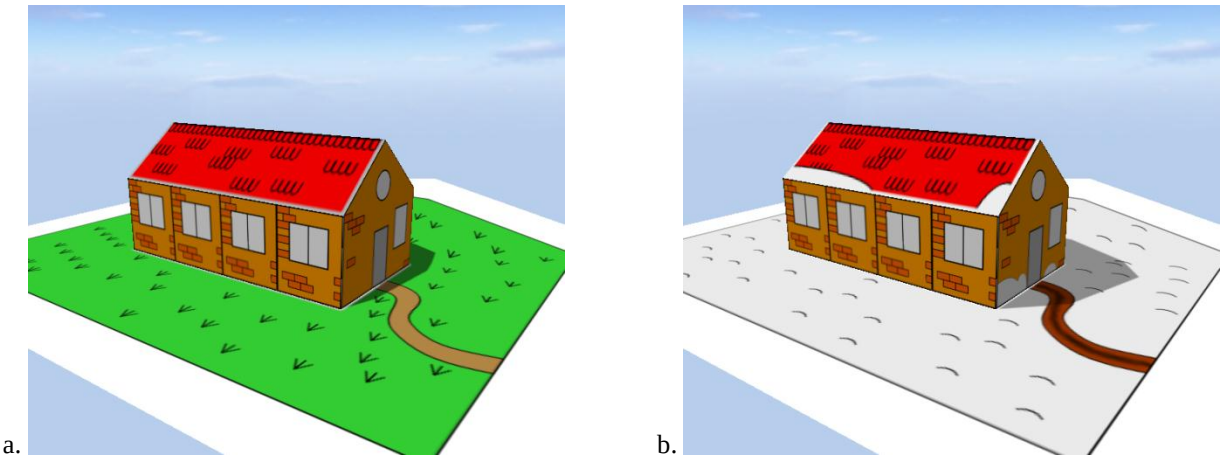

Fig. 83: Visualisation of a simple building in LOD2 using CityGML's appearance model. Two themes are defined for the building and the surrounding terrain: (a) theme showing the building in summer and (b) showing the building in winter (image: Hasso-Plattner-Institute).

```
<?xml version="1.0" encoding="utf-8"?>
<CityModel
  xmlns="http://www.opengis.net/citygml/2.0"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:app="http://www.opengis.net/citygml/appearance/2.0"
  xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```



```
xsi:schemaLocation="http://www.opengis.net/citygml/building/2.0 http://schemas.opengis.net/citygml/building/2.0/building.xsd
                    http://www.opengis.net/citygml/relief/2.0 http://schemas.opengis.net/citygml/relief/2.0/relief.xsd
                    http://www.opengis.net/citygml/appearance/2.0 http://schemas.opengis.net/citygml/appearance/2.0/appearance.xsd">
  <gml:boundedBy>
    <gml:Envelope srsDimension="3" srsName="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783">
      <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
      <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <cityObjectMember>
    <bldg:Building gml:id="GML_7b1a5a6f-ddad-4c3d-a507-3eb9ee0a8e68">
      <bldg:function
          codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_function.xml">1000</bldg:function>
      <bldg:yearOfConstruction>1985</bldg:yearOfConstruction>
      <bldg:roofType
          codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml">1030</bldg:roofType>
      <bldg:measuredHeight uom="#m">5.0</bldg:measuredHeight>
      <bldg:storeysAboveGround>1</bldg:storeysAboveGround>
      <bldg:storeyHeightsAboveGround uom="#m">3.0</bldg:storeyHeightsAboveGround>
      <bldg:lod1Solid>
        <gml:Solid>
          <gml:exterior>
            <gml:CompositeSurface gml:id="lod1Surface">
              <gml:surfaceMember>
                <gml:Polygon>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:posList srsDimension="3">458875 5438350 112 458885 5438350 112 458885 5438350 116 458875
                        5438350 116 458875 5438350 112 </gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
              …
              <gml:surfaceMember>
                <gml:Polygon gml:id="lod1RoofPoly1">
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:posList srsDimension="3">458875 5438350 116 458885 5438350 116 458885 5438355 116 458875
                        5438355 116 458875 5438350 116 </gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
              …
            </gml:CompositeSurface>
          </gml:exterior>
        </gml:Solid>
      </bldg:lod1Solid>
      <bldg:lod2Solid>
        <gml:Solid>
          <gml:exterior>
            <gml:CompositeSurface>
              <gml:surfaceMember>
                <gml:CompositeSurface gml:id="fLeft">
                  <gml:surfaceMember>
                    <gml:Polygon>
                      <gml:exterior>
                        <gml:LinearRing gml:id="fLeftExt1">
                          <gml:posList srsDimension="3">458875 5438350 112 458880 5438350 112 458880 5438350 115 458875
                            5438350 115 458875 5438350 112 </gml:posList>
                        </gml:LinearRing>
                      </gml:exterior>
                    </gml:Polygon>
                  </gml:surfaceMember>
                  <gml:surfaceMember>
                    <gml:Polygon>
                      <gml:exterior>
                        <gml:LinearRing gml:id="fLeftExt2">
                          <gml:posList srsDimension="3">458880 5438350 112 458885 5438350 112 458885 5438350 115 458880
                            5438350 115 458880 5438350 112 </gml:posList>
                        </gml:LinearRing>
                      </gml:exterior>
                    </gml:Polygon>
                  </gml:surfaceMember>
                </gml:CompositeSurface>
```

```
            </gml:surfaceMember>
            <gml:surfaceMember>
              <gml:Polygon gml:id="fFront">
                <gml:exterior>
                  <gml:LinearRing gml:id="fFrontExt">
                    <gml:posList srsDimension="3">458885 5438350 112 458885 5438355 112 458885 5438355 115 458885
                      5438352.5 117 458885 5438350 115 458885 5438350 112 </gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
            <gml:surfaceMember>
              <gml:Polygon gml:id="fRight">
                <gml:exterior>
                  <gml:LinearRing gml:id="fRightExt">
                    <gml:posList srsDimension="3">458885 5438355 112 458875 5438355 112 458875 5438355 115 458885
                      5438355 115 458885 5438355 112 </gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
            <gml:surfaceMember>
              <gml:Polygon gml:id="fBack">
                <gml:exterior>
                  <gml:LinearRing gml:id="fBackExt">
                    <gml:posList srsDimension="3">458875 5438355 112 458875 5438350 112 458875 5438350 115 458875
                      5438352.5 117 458875 5438355 115 458875 5438355 112 </gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
            <gml:surfaceMember>
              <gml:Polygon gml:id="lod2RoofPoly1">
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList srsDimension="3">458875 5438350 115 458885 5438350 115 458885 5438352.5 117 458875
                      5438352.5 117 458875 5438350 115 </gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
            <gml:surfaceMember>
              <gml:Polygon gml:id="lod2RoofPoly2">
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList srsDimension="3">458885 5438355 115 458875 5438355 115 458875 5438352.5 117 458885
                      5438352.5 117 458885 5438355 115 </gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList srsDimension="3">458875 5438350 112 458875 5438355 112 458885 5438355 112 458885
                      5438350 112 458875 5438350 112 </gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
          </gml:CompositeSurface>
        </gml:exterior>
      </gml:Solid>
    </bldg:lod2Solid>
    <bldg:address>
      … (address see LOD1 example)
    </bldg:address>
  </bldg:Building>
</cityObjectMember>
<cityObjectMember>
  <dem:ReliefFeature gml:id="GML_6bb30328-7599-4500-90ef-766fde6aa67b ">
    <dem:lod>1</dem:lod>
    <dem:reliefComponent>
      <dem:TINRelief gml:id=" GML_4eb161b0-aa7e-4087-937c-5c4c427c7fc9">
        <gml:name>Ground</gml:name>
```

```
          <dem:lod>1</dem:lod>
          <dem:tin>
            <gml:TriangulatedSurface gml:id="ground">
              <gml:trianglePatches>
                <gml:Triangle>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:posList>458868 5438362 112 458875 5438355 112 458883 5438362 114 458868 5438362 112
                      </gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Triangle>
                …
              </gml:trianglePatches>
            </gml:TriangulatedSurface>
          </dem:tin>
        </dem:TINRelief>
      </dem:reliefComponent>
    </dem:ReliefFeature>
  </cityObjectMember>
  <app:appearanceMember>
    <app:Appearance>
      <app:theme>Summer</app:theme>
      <app:surfaceDataMember>
        <app:X3DMaterial gml:id="lod1Material">
          <app:diffuseColor>1.0 0.6 0.0</app:diffuseColor>
          <app:target>#lod1Surface</app:target>
        </app:X3DMaterial>
      </app:surfaceDataMember>
      <app:surfaceDataMember>
        <app:GeoreferencedTexture>
          <app:imageURI>ground_summer.png</app:imageURI>
          <app:wrapMode>none</app:wrapMode>
          <app:referencePoint>
            <gml:Point>
              <gml:pos srsDimension="2">458870 5438360</gml:pos>
            </gml:Point>
          </app:referencePoint>
          <app:orientation>0.05 0.0 0.0 -0.05</app:orientation>
          <app:target>#ground</app:target>
          <app:target>#lod1RoofPoly1</app:target>
          <app:target>#lod2RoofPoly1</app:target>
          <app:target>#lod2RoofPoly2</app:target>
        </app:GeoreferencedTexture>
      </app:surfaceDataMember>
      <app:surfaceDataMember>
        <app:ParameterizedTexture gml:id="sideTexture">
          <app:imageURI>facade.png</app:imageURI>
          <app:wrapMode>wrap</app:wrapMode>
          <app:target uri="#fLeft">
            <app:TexCoordList>
              <app:textureCoordinates ring="#fLeftExt1">0.0 0.0 2.0 0.0 2.0 1.0 0.0 1.0 0.0 0.0</app:textureCoordinates>
              <app:textureCoordinates ring="#fLeftExt2">2.0 0.0 4.0 0.0 4.0 1.0 2.0 1.0 2.0 0.0</app:textureCoordinates>
            </app:TexCoordList>
          </app:target>
          <app:target uri="#fRight">
            <app:TexCoordGen>
              <app:worldToTexture>-0.4 0.0 0.0 183550.0 0.0 0.0 0.3333 -37.3333 0.0 0.0 0.0 1.0</app:worldToTexture>
            </app:TexCoordGen>
          </app:target>
        </app:ParameterizedTexture>
      </app:surfaceDataMember>
      <app:surfaceDataMember>
        <app:ParameterizedTexture>
          <app:imageURI>front_back_summer.png</app:imageURI>
          <app:wrapMode>none</app:wrapMode>
          <app:target uri="#fFront">
            <app:TexCoordList gml:id="frontTexCoord">
              <app:textureCoordinates ring="#fFrontExt">0.0 0.0 0.5 0.0 0.5 0.6 0.25 1.0 0.0 0.6 0.0
                0.0</app:textureCoordinates>
            </app:TexCoordList>
          </app:target>
          <app:target uri="#fBack">
            <app:TexCoordList gml:id="backTexCoord">
              <app:textureCoordinates ring="#fBackExt">0.5 0.0 1.0 0.0 1.0 0.6 0.75 1.0 0.5 0.6 0.5
                0.0</app:textureCoordinates>
```

```
            </app:TexCoordList>
          </app:target>
        </app:ParameterizedTexture>
      </app:surfaceDataMember>
    </app:Appearance>
  </app:appearanceMember>
  <app:appearanceMember>
    <app:Appearance>
      <app:theme>Winter</app:theme>
      <app:surfaceDataMember>
        <app:GeoreferencedTexture>
          <app:imageURI>ground_winter.png</app:imageURI>
          <app:wrapMode>none</app:wrapMode>
          <app:referencePoint>
            <gml:Point>
              <gml:pos srsDimension="2">458870 5438360</gml:pos>
            </gml:Point>
          </app:referencePoint>
          <app:orientation>0.05 0.0 0.0 -0.05</app:orientation>
          <app:target>#ground</app:target>
          <app:target>#lod1RoofPoly1</app:target>
          <app:target>#lod2RoofPoly1</app:target>
          <app:target>#lod2RoofPoly2</app:target>
        </app:GeoreferencedTexture>
      </app:surfaceDataMember>
      <app:surfaceDataMember xlink:href="#lod1Material"/>
      <app:surfaceDataMember xlink:href="#sideTexture"/>
      <app:surfaceDataMember>
        <app:ParameterizedTexture>
          <app:imageURI>front_back_winter.png</app:imageURI>
          <app:wrapMode>none</app:wrapMode>
          <app:target uri="#fFront" xlink:href="#frontTexCoord"/>
          <app:target uri="#fBack" xlink:href="#backTexCoord"/>
        </app:ParameterizedTexture>
      </app:surfaceDataMember>
    </app:Appearance>
  </app:appearanceMember>
</CityModel>
```

Listing 9: Excerpt from the CityGML dataset illustrating CityGML's appearance model. The dataset is visualised in Fig. 82 and Fig. 83.

The following three raster images (Fig. 84 - Fig. 86) are referenced in the dataset by *ParameterizedTexture* objects to texturize the vertical boundary surfaces of the building in LOD2. The image *facade.png* (cf. Fig. 84) is assigned to the side surfaces using the texture wrapping mode *wrap* and is applied both within the summer and the winter theme.

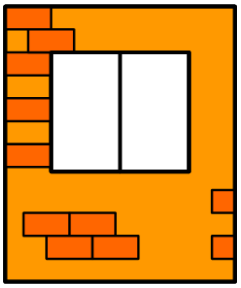

Fig. 84: Image *facade.png* used in the dataset to texturize the side surfaces of the building in LOD2 (cf. Fig. 83 a. and b.) (image: Hasso-Plattner-Institute).

Fig. 85 shows the texture atlas *front_back_summer.png* combining the textures for the front surface and the back surface of the building in LOD2 within the summer theme. Only a portion of this image is assigned to the specific surfaces. The relevant parts are defined using a *TextCoordList* object.

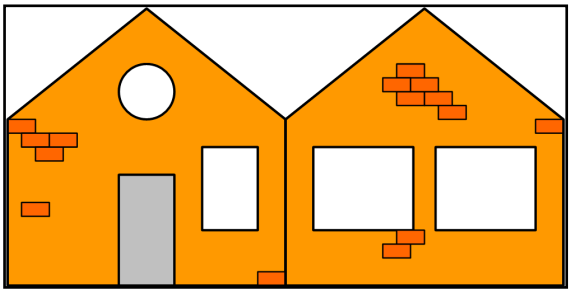

Fig. 85: Texture atlas *front_back_summer.png* containing the textures for the front surface and the back surface of the building in LOD2 within the summer theme (cf. Fig. 83 a.) (image: Hasso-Plattner-Institute).

Identically to *front_back_summer.png* the texture atlas *front_back_winter.png* contains the textures for the front surface and the back surface of the building in LOD2 within the winter theme.

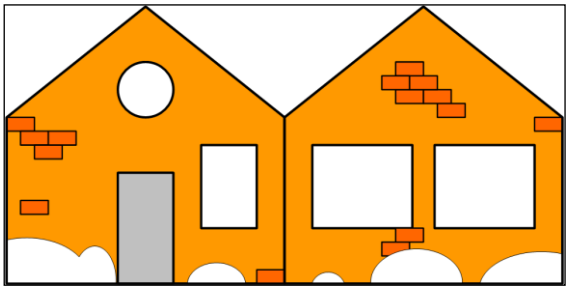

Fig. 86: Texture atlas *front_back_winter.png* containing the textures for the front surface and the back surface of the building in LOD2 within the winter theme (cf. Fig. 83 b.) (image: Hasso-Plattner-Institute).

The raster images shown in Fig. 87 and Fig. 88 are assigned to the terrain and the roof surfaces of the building in LOD1 as well as in LOD2. In the dataset this is implemented by a *GeoreferencedTexture* object linking to the according GML geometry objects. Whereas the image *ground_summer.png* (cf. Fig. 87) represents the texture for the summer theme, *ground_winter.png* (cf. Fig. 88) is used within the winter theme.

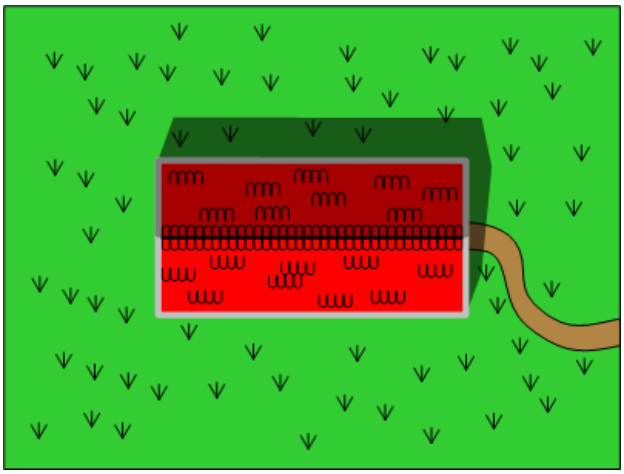

Fig. 87: The image *ground_summer.png* is assigned to the terrain and the roof surfaces of the building both in LOD1 and LOD2 (cf. Fig. 82 a. and Fig. 83 a.) within the summer theme (image: Hasso-Plattner-Institute).

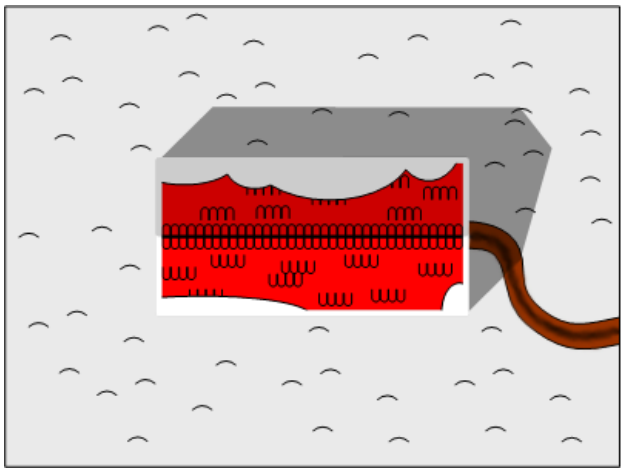

Fig. 88: The image *ground_winter.png* is assigned to the terrain and the roof surfaces of the building both in LOD1 and LOD2 (cf. Fig. 82 b. and Fig. 83 b.) within the winter theme (image: Hasso-Plattner-Institute).

Instead of denoting the georeference of the *GeoreferencedTexture* elements inline as shown in Listing 9, the *Appearance* module also supports to provide the texture mapping with the image file (e.g. using a georeferenced TIFF) or as a separate ESRI world file. The latter is a six-line text file with a decimal number on each line. These numbers correspond to the parameters of the inline texture mapping and can be 1:1 copied from there (and vice versa). The rotation and scaling parameters given by the *orientation* property of *GeoreferencedTexture* (2x2 matrix represented as a list of 4 doubles in row-major) are given on lines 1, 3, 2, and 4 in an ESRI world file. And the location of the center of the upper left image pixel in world space denoted by the *referencePoint* property of *GeoreferencedTexture* corresponds to lines 5 and 6. For the example in Listing 9, the world files for both texture images contain the following values:

```
0.05
0.0
0.0
-0.05
458870
5438360
```

Listing 10: Contents of a world file specifying the same texture mapping for a *GeoreferencedTexture* as given by the inline georeference shown in Listing 9.

In order to make use of this world file, it has to be stored together with the texture image (for filename conventions, see http://en.wikipedia.org/wiki/World_file). Additionally, the inline texture mapping has to be removed from both *GeoreferencedTexture* elements in Listing 9 as shown in the following excerpt. Both alternatives for denoting the georeference of a *GeoreferencedTexture* are bundled with the CityGML schemas, and are available at http://schemas.opengis.net/citygml/examples/2.0/appearance/.

```
…
  <app:GeoreferencedTexture>
    <app:imageURI>ground_winter.png</app:imageURI>
    <app:target>#ground</app:target>
    <app:target>#lod1RoofPoly1</app:target>
    <app:target>#lod2RoofPoly1</app:target>
    <app:target>#lod2RoofPoly2</app:target>
  </app:GeoreferencedTexture>
…
```

Listing 11: Adapted *GeoreferencedTexture* element for which the texture mapping is provided by the world file shown in Listing 10.

G.8 Example of a CityGML dataset illustrating the use of texture coordinates for complex surfaces with holes

The following example demonstrates the use of texture coordinates for complex surfaces with holes. Additionally, the concept of overwriting (cf. chapter 9.1) is exemplified. The example shows a textured road in LOD1, which consists of a roundabout and two entering streets (Fig. 89). It uses two *app:ParameterizedTexture* objects, a regular road piece (`rd`, Fig. 90) and a dirt track becoming a paved road (`dt`, Fig. 91). All geometry is contained in a single *gml:MultiSurface* (`road`) being the target of `rd`. The roundabout is modelled as a polygon with holes (`roundaboutPoly`). Both rings `raEx` and `raIn` need to receive texture coordinates. For efficient texturing, texture coordinates and wrap mode are chosen such that the regular road piece is draped onto the polygonal ring as desired at the price of some distortion. A distortion-free alternative, e.g. for higher LODs, requires an additional road piece texture for the roundabout (Fig. 92) or a texture for the complete roundabout (Fig. 93). In both cases, texture space is wasted. Wasted areas are marked red.

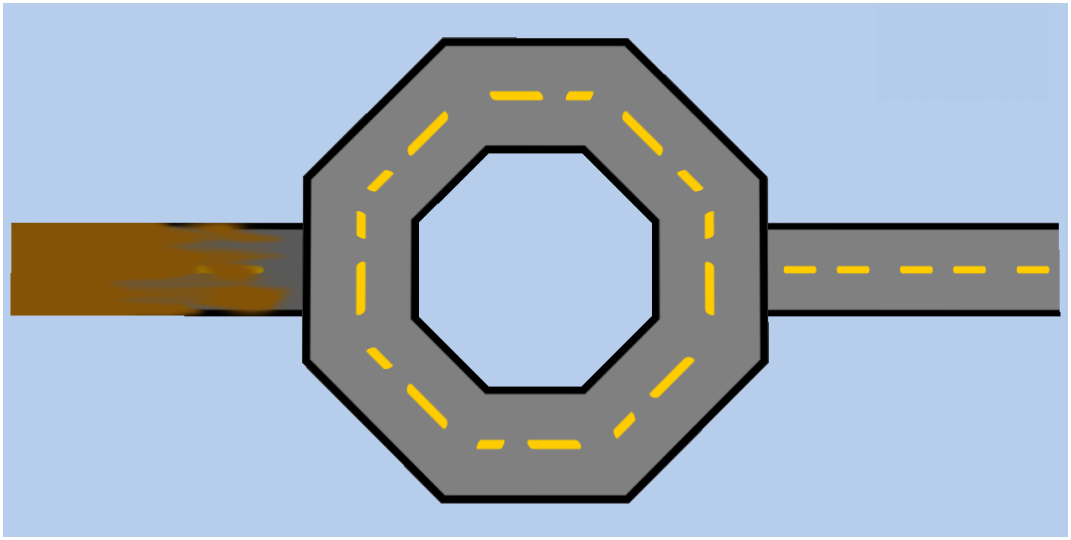

Fig. 89: A rendering of the textured geometry (images for this example: Hasso-Plattner-Institute).

Fig. 90: The road piece texture.

Fig. 91: The dirt track texture.

Fig. 92: A distortion-free road piece for the roundabout.
Red areas never become visible and are wasted.

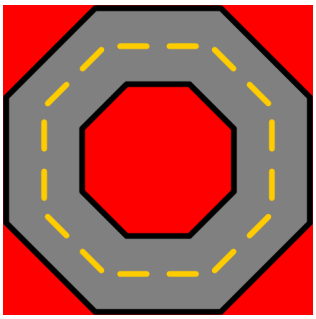

Fig. 93: The complete roundabout in a distortion-free texture.
Red areas never become visible and are wasted.

The entering dirt track requires a different texture. Even though its geometry (`dirtPoly`) is contained in the already textured `road`, the existing texture is overwritten and replaced by using `dirtPoly` as target of `dt`. Overwriting only occurs when assigning a new texture to a surface geometry object contained in an already textured aggregated geometry object. Assigning two textures to the same surface geometry object directly is not allowed. The same applies to materials.

```
<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns="http://www.opengis.net/citygml/2.0"
  xmlns:app="http://www.opengis.net/citygml/appearance/2.0"
  xmlns:tran="http://www.opengis.net/citygml/transportation/2.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/citygml/appearance/2.0 http://schemas.opengis.net/citygml/appearance/2.0/appearance.xsd
        http://www.opengis.net/citygml/transportation/2.0 http://schemas.opengis.net/citygml/transportation/2.0/transportation.xsd">
  <gml:boundedBy>
    <!--The srsName attribute references a local engineering CRS that is defined in the example dataset provided in annex G.9 -->
    <gml:Envelope srsDimension="3" srsName="local-CRS-1">
      <gml:lowerCorner>-45.0 -20.0 0.0</gml:lowerCorner>
      <gml:upperCorner>45.0 20.0 10.0</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <cityObjectMember>
    <tran:Road>
      <app:appearance>
        <app:Appearance>
          <app:theme>visual</app:theme>
          <app:surfaceDataMember>
            <app:ParameterizedTexture gml:id="rd">
              <app:imageURI>rd.png</app:imageURI>
              <app:wrapMode>mirror</app:wrapMode>
              <app:target uri="#road">
                <app:TexCoordList>
                  <app:textureCoordinates ring="#raEx"> 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 </app:textureCoordinates>
                  <app:textureCoordinates ring="#raIn"> 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 </app:textureCoordinates>
                  <app:textureCoordinates ring="#roadEx"> 0 0 2.5 0 2.5 1 0 1 0 0 </app:textureCoordinates>
                  <app:textureCoordinates ring="#dirtEx"> 0 0 2.5 0 2.5 1 0 1 0 0 </app:textureCoordinates>
                </app:TexCoordList>
              </app:target>
            </app:ParameterizedTexture>
          </app:surfaceDataMember>
          <app:surfaceDataMember>
            <app:ParameterizedTexture gml:id="dt">
              <app:imageURI>dt.png</app:imageURI>
              <app:wrapMode>mirror</app:wrapMode>
              <app:target uri="#dirtPoly">
                <app:TexCoordList>
                  <app:textureCoordinates ring="#dirtEx"> 0 0 1 0 1 1 0 1 0 0 </app:textureCoordinates>
                </app:TexCoordList>
              </app:target>
            </app:ParameterizedTexture>
          </app:surfaceDataMember>
        </app:Appearance>
      </app:appearance>
      <tran:lod1MultiSurface>
        <gml:MultiSurface gml:id="road">
          <gml:surfaceMember>
            <gml:Polygon gml:id="roundaboutPoly">
```

```
            <gml:exterior>
              <gml:LinearRing gml:id="raEx">
                <gml:posList srsDimension="3"> -8 20 5 -20 8 5 -20 -8 5 -8 -20 5 8 -20 5 20 -8 5 20 8 5 8 20 5
                  -8 20 5 </gml:posList>
              </gml:LinearRing>
            </gml:exterior>
            <gml:interior>
              <gml:LinearRing gml:id="raIn">
                <gml:posList srsDimension="3"> -4 10 5 4 10 5 10 4 5 10 -4 5 4 -10 5 -4 -10 5 -10 -4 5 -10 4 5
                  -4 10 5 </gml:posList>
              </gml:LinearRing>
            </gml:interior>
          </gml:Polygon>
        </gml:surfaceMember>
        <gml:surfaceMember>
          <gml:Polygon gml:id="roadPoly">
            <gml:exterior>
              <gml:LinearRing gml:id="roadEx">
                <gml:posList srsDimension="3"> 20 -4 5 45 -4 5 45 4 5 20 4 5 20 -4 5 </gml:posList>
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </gml:surfaceMember>
        <gml:surfaceMember>
          <gml:Polygon gml:id="dirtPoly">
            <gml:exterior>
              <gml:LinearRing gml:id="dirtEx">
                <gml:posList srsDimension="3"> -20 -4 5 -45 -4 5 -45 4 5 -20 4 5 -20 -4 5 </gml:posList>
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </gml:surfaceMember>
      </gml:MultiSurface>
    </tran:lod1MultiSurface>
  </tran:Road>
 </cityObjectMember>
</CityModel>
```

Listing 12: CityGML dataset illustrating the use of texture coordinates for complex surfaces with holes. The dataset is visualised in Fig. 89.

G.9 Example of a CityGML dataset illustrating the use of local coordinate reference systems

The following dataset demonstrates the use of local engineering coordinate reference systems in CityGML. The dataset is based on the LOD1 example given in annex G.2. The definition of the local CRS is encoded in the *gml:metaDataProperty* of the *CityModel* element and describes a 3D Cartesian reference system. Its origin corresponds to an anchor point which references a point on the earth's surface (here: in Germany). This anchor point is defined in the *EngineeringDatum* element that is used by the local CRS, and is associated with a compound 3D CRS which combines a Projected CRS (ETRS89 / UTM zone 32N; EPSG code 25832) for the planimetry with a Vertical CRS (DHHN92 height, EPSG code 5783) for the height reference. The *gml:Envelope* of the *CityModel* references the local CRS by its *gml:id "local-CRS-1"* and thus it is inherited by all geometries in all properties of the *CityModel* and members of this feature collection. If required, the reference system may be overridden by the members of the *CityModel*, for example, to provide a separate anchor point for each city object.

```
<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.opengis.net/citygml/2.0"
  xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xsi:schemaLocation="http://www.opengis.net/citygml/building/2.0 http://schemas.opengis.net/citygml/building/2.0/building.xsd
                      http://www.opengis.net/citygml/relief/2.0 http://schemas.opengis.net/citygml/relief/2.0/relief.xsd ">
  <gml:metaDataProperty>
    <!-- Local EngineeringCRS definition contained specified inline as metadata in this XML instance. -->
    <!-- This CRS is referenced by geometry throughtout this instance by srsName value #local-CRS-1 -->
    <gml:EngineeringCRS xmlns:metadata="urn:x-ogp:spec:schema-xsd:localmetadata" gml:id="local-CRS-1">
      <gml:metaDataProperty>
        <metadata:CommonMetaData>
          <metadata:type>engineering</metadata:type>
        </metadata:CommonMetaData>
      </gml:metaDataProperty>
      <gml:srsName codeSpace="XYZ">urn:ogc:def:crs:local:CRS:1</gml:srsName>
      <gml:scope>CityGML</gml:scope>
      <gml:usesCS>
        <gml:CartesianCS gml:id="local-CS-1">
          <gml:metaDataProperty>
            <metadata:CommonMetaData>
              <metadata:type>Cartesian</metadata:type>
              <metadata:description>Cartesian 3D CS. Axes: UoM: m.</metadata:description>
            </metadata:CommonMetaData>
          </gml:metaDataProperty>
          <gml:csName codeSpace="XYZ">urn:ogc:def:crs:local:CS:1</gml:csName>
          <gml:usesAxis>
            <gml:CoordinateSystemAxis gml:id="local-axis-1" gml:uom="urn:ogc:def:uom:EPSG::9001">
              <gml:name/>
              <gml:axisID>
                <gml:name>X</gml:name>
              </gml:axisID>
              <gml:axisAbbrev>x</gml:axisAbbrev>
              <gml:axisDirection codeSpace="XYZ">X</gml:axisDirection>
            </gml:CoordinateSystemAxis>
          </gml:usesAxis>
          <gml:usesAxis>
            <gml:CoordinateSystemAxis gml:id="local-axis-2" gml:uom="urn:ogc:def:uom:EPSG::9001">
              <gml:name/>
              <gml:axisID>
                <gml:name>Y</gml:name>
              </gml:axisID>
              <gml:axisAbbrev>y</gml:axisAbbrev>
              <gml:axisDirection codeSpace="XYZ">Y</gml:axisDirection>
            </gml:CoordinateSystemAxis>
          </gml:usesAxis>
```



```
            <gml:usesAxis>
              <gml:CoordinateSystemAxis gml:id="local-axis-3" gml:uom="urn:ogc:def:uom:EPSG::9001">
                <gml:name/>
                <gml:axisID>
                  <gml:name>Z</gml:name>
                </gml:axisID>
                <gml:axisAbbrev>z</gml:axisAbbrev>
                <gml:axisDirection codeSpace="XYZ">Z</gml:axisDirection>
              </gml:CoordinateSystemAxis>
            </gml:usesAxis>
          </gml:CartesianCS>
        </gml:usesCS>
        <gml:usesEngineeringDatum>
          <gml:EngineeringDatum gml:id="local-datum-1">
            <gml:metaDataProperty>
              <metadata:CommonMetaData>
                <metadata:type>Cartesian datum</metadata:type>
              </metadata:CommonMetaData>
            </gml:metaDataProperty>
            <gml:datumName codeSpace="XYZ">Datum1</gml:datumName>
            <gml:anchorPoint codeSpace="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783">
                458868.0 5438343.0 112.0</gml:anchorPoint>
            <!-- The anchor point defines the origin of the local CS with respect to the world CRS -->
            <!-- In this example, the anchor point references a point on the earth (in Germany) using a compound CRS -->
            <!-- For planimetry, the reference system ETRS89 / UTM zone 32N (EPSG code 25832) is used -->
            <!-- The vertical reference system is DHHN92 height (EPSG code 5783) -->
          </gml:EngineeringDatum>
        </gml:usesEngineeringDatum>
      </gml:EngineeringCRS>
    </gml:metaDataProperty>
    <gml:description>Simple example for a CityGML dataset using a local engineering CRS</gml:description>
    <gml:name>Simple 3D city model LOD1 without Appearance</gml:name>
    <gml:boundedBy>
      <gml:Envelope srsName="#local-CRS-1">
        <!-- Encoding of local-CRS-1 is specified in CityModel metadataProperty in this document-->
        <gml:pos srsDimension="3">0.0 0.0 0.0</gml:pos>
        <gml:pos srsDimension="3">24.0 19.0 4.0</gml:pos>
      </gml:Envelope>
    </gml:boundedBy>
    <cityObjectMember>
      <bldg:Building gml:id="GML_7b1a5a6f-ddad-4c3d-a507-3eb9ee0a8e68">
        <gml:name>Example Building LOD1 </gml:name>
        … (Attributes see example LOD1)
        <bldg:lod1Solid>
          <gml:Solid>
            <gml:exterior>
              <gml:CompositeSurface gml:id="lod1Surface">
                <!-- Face Side 1 -->
                <gml:surfaceMember>
                  <gml:Polygon>
                    <gml:exterior>
                      <gml:LinearRing>
                        <gml:posList>7.0 7.0 0.0 17.0 7.0 0.0 17.0 7.0 4.0 7.0 7.0 4.0 7.0 7.0 0.0</gml:posList>
                      </gml:LinearRing>
                    </gml:exterior>
                  </gml:Polygon>
                </gml:surfaceMember>
                <!-- Face Side 2 -->
                <gml:surfaceMember>
                  <gml:Polygon>
                    <gml:exterior>
                      <gml:LinearRing>
                        <gml:posList>17.0 7.0 0.0 17.0 12.0 0.0 17.0 12.0 4.0 17.0 7.0 4.0 17.0 7.0 0.0</gml:posList>
                      </gml:LinearRing>
                    </gml:exterior>
                  </gml:Polygon>
                </gml:surfaceMember>
                <!-- Face Side 3 -->
                <gml:surfaceMember>
```

```
                <gml:Polygon>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:posList>17.0 12.0 0.0 7.0 12.0 0.0 7.0 12.0 4.0 17.0 12.0 4.0 17.0 12.0 0.0</gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
              <!-- Face Side 4 -->
              <gml:surfaceMember>
                <gml:Polygon>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:posList>7.0 12.0 0.0 7.0 7.0 0.0 7.0 7.0 4.0 7.0 12.0 4.0 7.0 12.0 0.0</gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
              <!-- Face Top -->
              <gml:surfaceMember>
                <gml:Polygon>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:posList>7.0 7.0 4.0 17.0 7.0 4.0 17.0 12.0 4.0 7.0 12.0 4.0 7.0 7.0 4.0</gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
              <!-- Face Bottom -->
              <gml:surfaceMember>
                <gml:Polygon>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:posList>7.0 7.0 0.0 7.0 12.0 0.0 17.0 12.0 0.0 17.0 7.0 0.0 7.0 7.0 0.0</gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:CompositeSurface>
          </gml:exterior>
        </gml:Solid>
      </bldg:lod1Solid>
      <bldg:address>
        <Address>
        …
        </Address>
      </bldg:address>
    </bldg:Building>
  </cityObjectMember>
  <cityObjectMember>
    <dem:ReliefFeature gml:id="GML_6bb30328-7599-4500-90ef-766fde6aa67b">
      <gml:name>Example TIN LOD1</gml:name>
      <dem:lod>1</dem:lod>
      <dem:reliefComponent>
        <dem:TINRelief gml:id="GUID_04D4DsNGv1MfvYu5O3lkcW">
          <gml:name>Ground</gml:name>
          <dem:lod>1</dem:lod>
          <dem:tin>
            <gml:TriangulatedSurface gml:id="ground">
              <gml:trianglePatches>
                <gml:Triangle>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:posList>0.0 19.0 0.0 7.0 12.0 0.0 15.0 19.0 2.0 0.0 19.0 0.0</gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Triangle>
                <gml:Triangle>
                  …
```

```
                </gml:Triangle>
                  ... (more triangles)
              </gml:trianglePatches>
            </gml:TriangulatedSurface>
          </dem:tin>
        </dem:TINRelief>
      </dem:reliefComponent>
    </dem:ReliefFeature>
  </cityObjectMember>
</CityModel>
```

Listing 13: CityGML dataset illustrating the use of local coordinate reference systems. The dataset is visualised in Fig. 76 on page 272.

Annex H
(informative)

Example ADE for Noise Immission Simulation

This annex illustrates the usage of CityGML within an environmental simulation application. The definition of the corresponding Application Domain Extension (ADE) is included as an example.

The Environmental Noise Directive of the European Union 2002/49/EG obligates the EU member states to calculate every 5 years the noise levels at a height of 4m on buildings and to document the results in noise maps. The noise maps serve as information for the European Union and the citizens affected by noise (Fig. 94). These noise maps are generated on the basis of acoustic models and noise propagation calculations, not on the basis of measurements (Fig. 95). For the noise propagation calculations, a great number of thematic data and 3D geodata is necessary for each EU member state. Because of the large spatial extent of the noise calculation, the provision of statewide and ubiquitous 3D geodata on buildings, roads, railways and terrain for a multitude of users is necessary, in part with high requirements on resolution.

The calculation of noise levels from a road requires information about the traffic flow, the heavy vehicle percentage, the speed limits, the road surface types and the road gradient. Furthermore, the noise level depends on the distance between point of emission and reception (immission) as well as on reflection (e.g. on building facades) or shielding effects (e.g. noise barriers). The noise level is calculated separately for the day (06.00-18.00), the evening (18.00-22.00) and the night (22.00-06.00). As the noise level is calculated at a height of 4m and as the influence of vertical reflecting surfaces is considered (e.g. noise barriers and buildings), a multitude of geodata in the third dimension is necessary. In addition to all 3D geodata specific thematic data are required. For example the following data are necessary for the noise calculation of roads: Digital Terrain Model with 10m grid, 3D building models with their thematic attributes (e.g. reflection, inhabitants), 3D road data with their thematic attributes (e.g. traffic flow, heavy vehicle percentage, speed limit, type of road surface, road gradient, width of a road), 3D noise barriers and their thematic attributes (e.g. reflection).

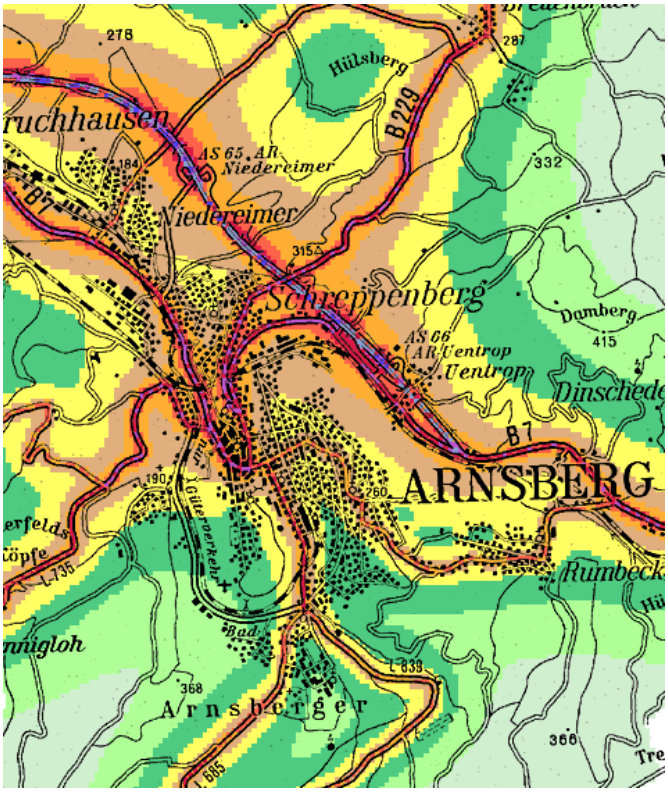

Fig. 94: Noise map generated from the 3D CityGML geodata and used for the reporting demanded of the EU Environmental Noise Directive (dark colours show higher noise immission) (source: State Agency for nature, environment and consumer protection NRW).

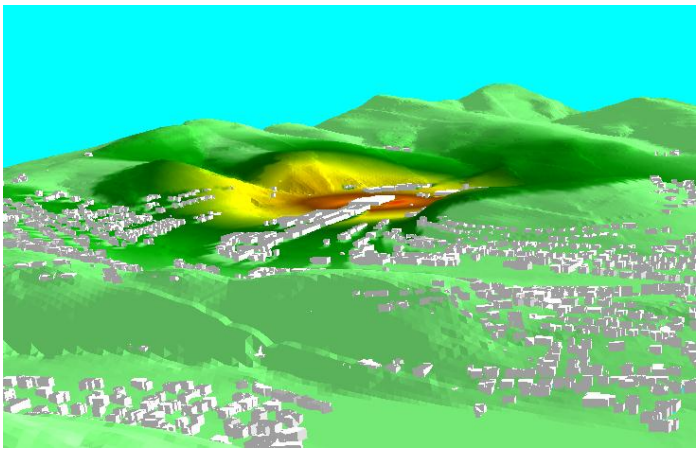

Fig. 95: Modelling a noise emission source using the 3D CityGML geodata in a special noise calculation software as first step before generating the noise map in Fig. 94 (source: Surveying and Mapping Agency NRW, Stapelfeldt GmbH, Institute of Geodesy and Geoinformation Uni Bonn).

In the state of North Rhine-Westphalia, special conditions have to be considered: high population and transportation route density and therefore the highest amount of noise calculation areas and objects in Germany. The aim is to provide a sustainable, efficient and variable access to the required 3D geodata for the 5 years iteration period and the different noise calculation authorities.

In order to provide this considerable amount of statewide 3D geodata, the responsible partners, in particular the State Ministry of Environment, Nature Conservation, Agriculture and Consumer Protection of North Rhine-Westphalia, the State Agency for nature, environment and consumer protection of North Rhine-Westphalia and the Surveying and Mapping Agency of North Rhine-Westphalia, have decided to use the Spatial Data Infrastructure in North Rhine-Westphalia (GDI NRW) and to extend it with statewide Web Services for 2.5D and 3D geoinformation. Therefore, new OGC Web Services for building models, terrain, road and railway data were implemented (e.g. Web Feature Service for 3D block models in LOD1 and for 3D road and railway data, Web Coverage Service for DTM).

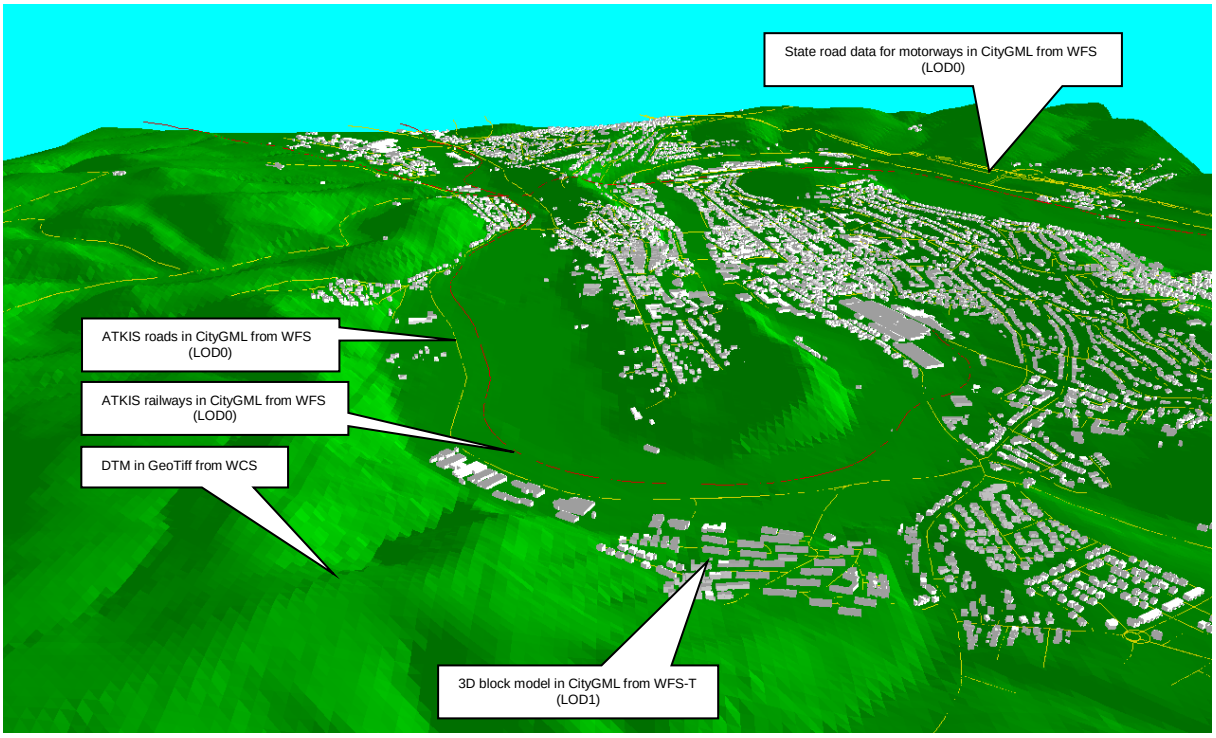

Fig. 96: 3D geodata in CityGML for the calculation of the noise map in Fig. 94: DTM in GeoTiff, 3D block model in CityGML, 3D road and railway data in CityGML, state road data for higher-level roads in CityGML (source: Surveying and Mapping Agency NRW, State Road Enterprice NRW, Stapelfeldt GmbH, Institute of Geodesy and Geoinformation Uni Bonn).

CityGML is used together with GeoTIFF as the only exchange formats between web services and noise calculation software (Fig. 96 -Fig. 98). For the special requirements of the noise directive, a CityGML noise application schema has been developed by the Institute of Geodesy and Geoinformation University of Bonn and the Special Interest Group SIG 3D of GDI NRW. It is based on the ADE mechanism (see chapter 6.12 and 10.13). This mechanism allows the supplementation of existing classes and objects in CityGML (e.g. buildings) by thematic attributes. The quantity as well as the type of these attributes is selectable. The CityGML schema can also be complemented by new classes. Hence, the noise application schema contains new objects (e.g. segmentation of roads according to noise requirements - *NoiseRoadSegment*, Fig. 99) as well as noise attributes attached to existing objects (e.g. reflection of buildings, Fig. 100). These additional noise attributes are derived from regulations issued by the Federal Government of Germany realising the obligations of the Environmental Noise Directive of the European Union (cf. BImSchV 2006, VBUS 2006, VBUSch 2006).

The interoperability techniques of this project demonstrate a remarkable innovation, as for the first time statewide 3D geodata are provided via common standards and web services (cf. Czerwinski et al. 2006b). Therefore, the Spatial Data Infrastructure for noise calculation in North Rhine-Westphalia provides an application example for the INSPIRE directive of the European Union 2007/2/EC (Infrastructure for Spatial Information in Europe) (cf. Czerwinski et al. 2007).

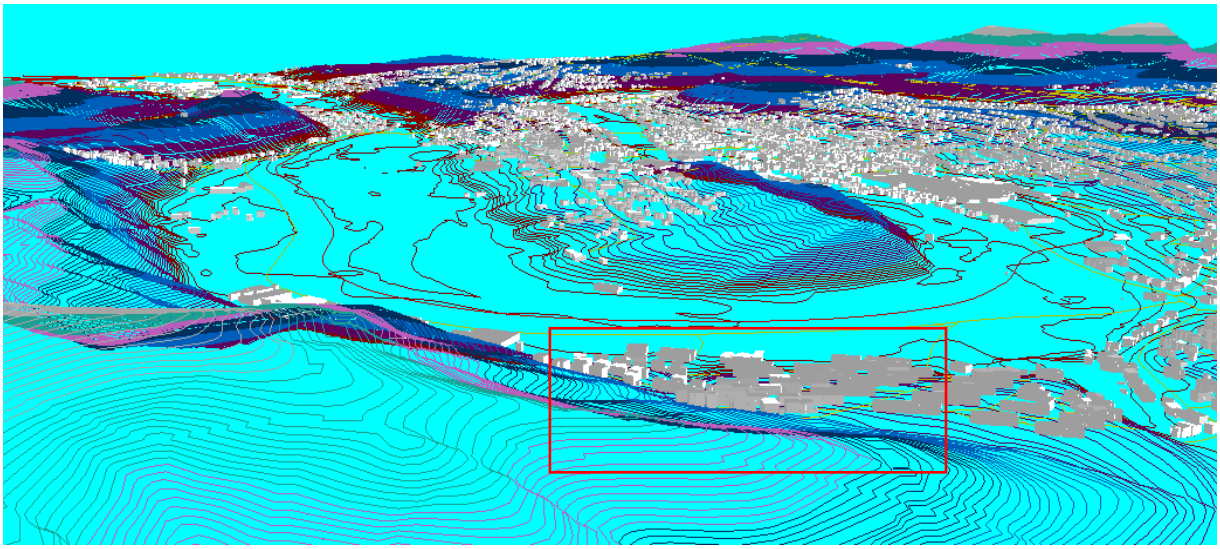

Fig. 97: 3D geodata in CityGML for the calculation of the noise map in Fig. 94: Derived contour lines for the generation of CityGML breaklines, 3D block model in CityGML, 3D road and railway data in CityGML, state road data for higher-level roads in CityGML (source: Surveying and Mapping Agency NRW, State Road Enterprise NRW, Stapelfeldt GmbH, Institute of Geodesy and Geoinformation Uni Bonn).

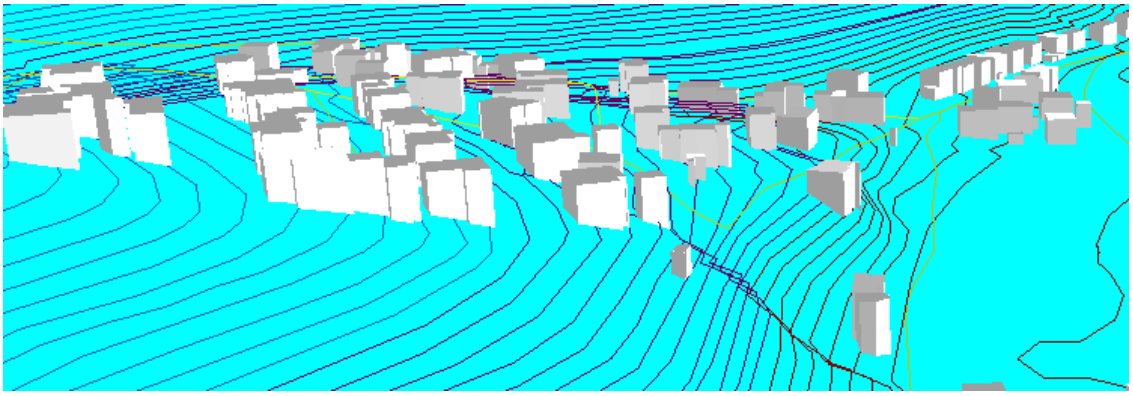

Fig. 98: Extract from Fig. 97 shows the integration of 3D block models in the DTM by appropriate CityGML modelling (lowest point of ALK building polygon is taken as measure to generate the building bottom side) (source: Surveying and Mapping Agency NRW, Stapelfeldt GmbH, Institute of Geodesy and Geoinformation Uni Bonn).

H.1 CityGML Noise ADE

In this section the data models for the CityGML Noise ADE are given as UML diagrams and XML schema. As the semantics of the specific attributes and object types result from the German regulations for noise immision computations, they are not explained in detail here (see BimSchV 2006, VBUS 2006, VBUSch 2006). The purpose of this section is to provide an example how CityGML can be extended using the ADE mechanism. The XML Schema definition of the CityGML Noise ADE together with example datasets can be obtained from http://schemas.opengis.net/citygml/examples/2.0/ade/noise-ade/.

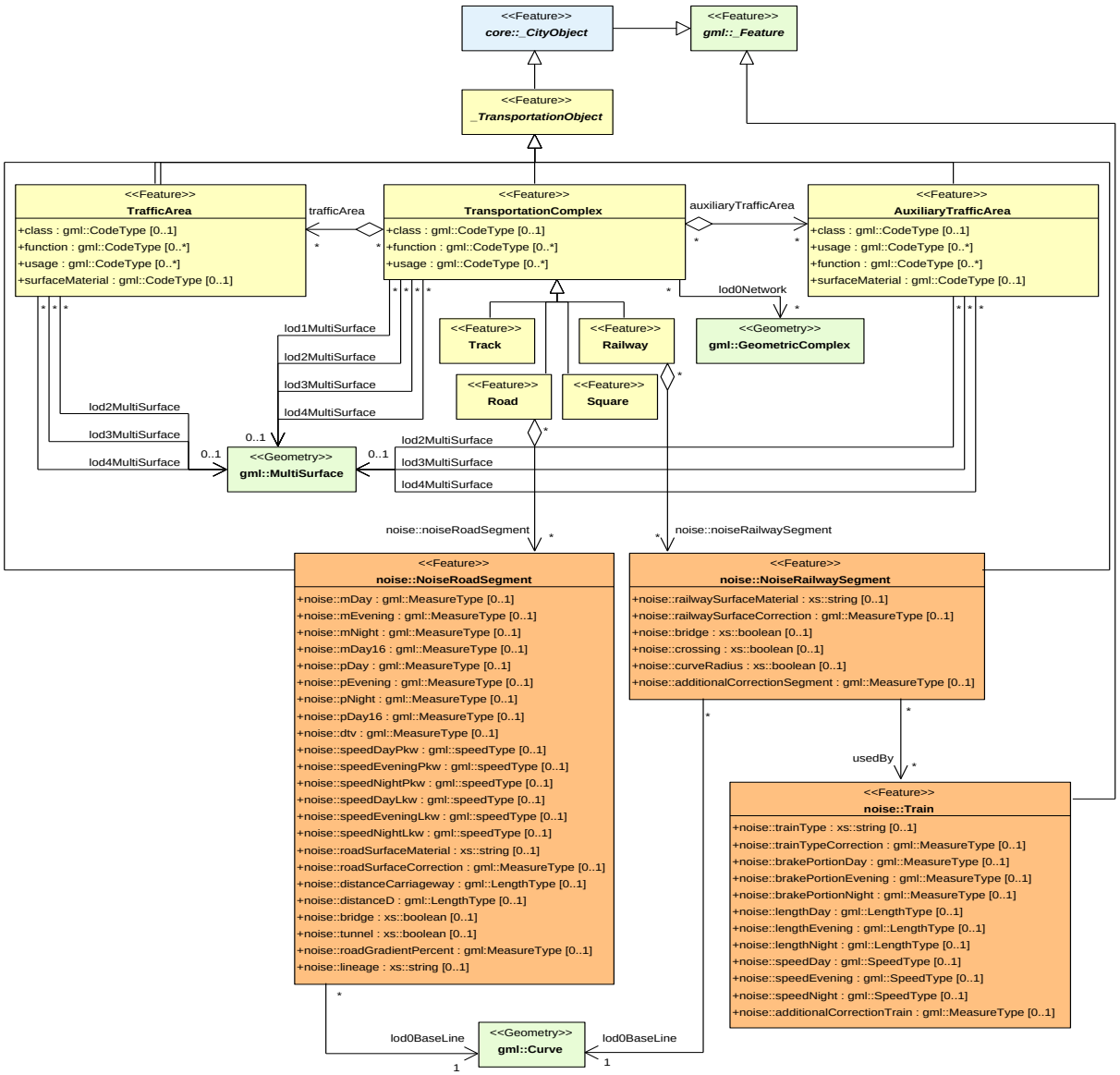

Fig. 99: CityGML noise application schema – transportation model (light yellow=CityGML *Transportation* module, light orange=CityGML Noise ADE). Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Transportation* module. The prefix *noise* is associated with the CityGML Noise ADE (source: Institute of Geodesy and Geoinformation Uni Bonn).

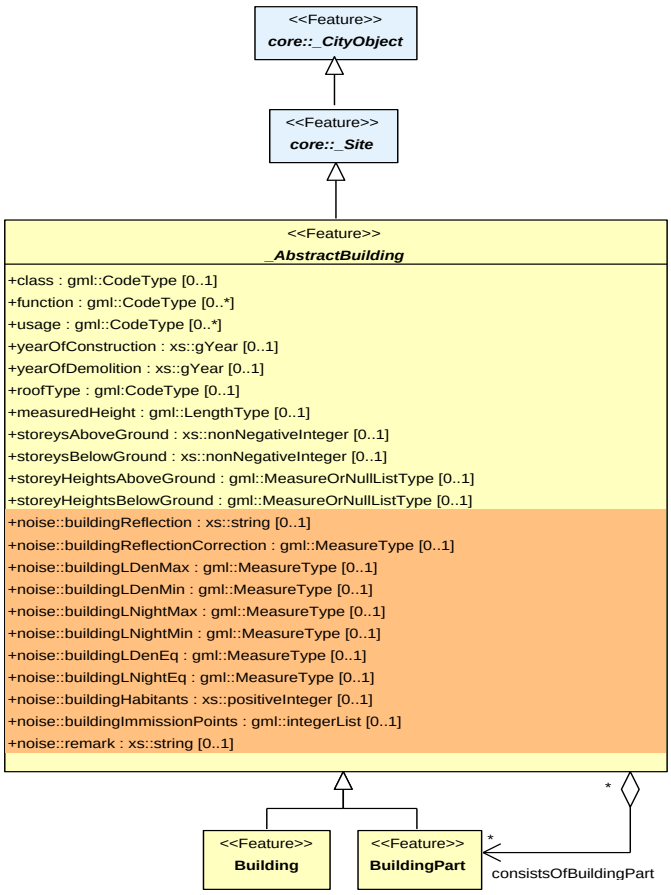

Fig. 100: CityGML noise application schema – excerpt of the building model (light yellow=CityGML *Building* module, light orange=CityGML Noise ADE). Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Building* module. The prefix *noise* is associated with the CityGML Noise ADE (source: Institute of Geodesy and Geoinformation Uni Bonn).

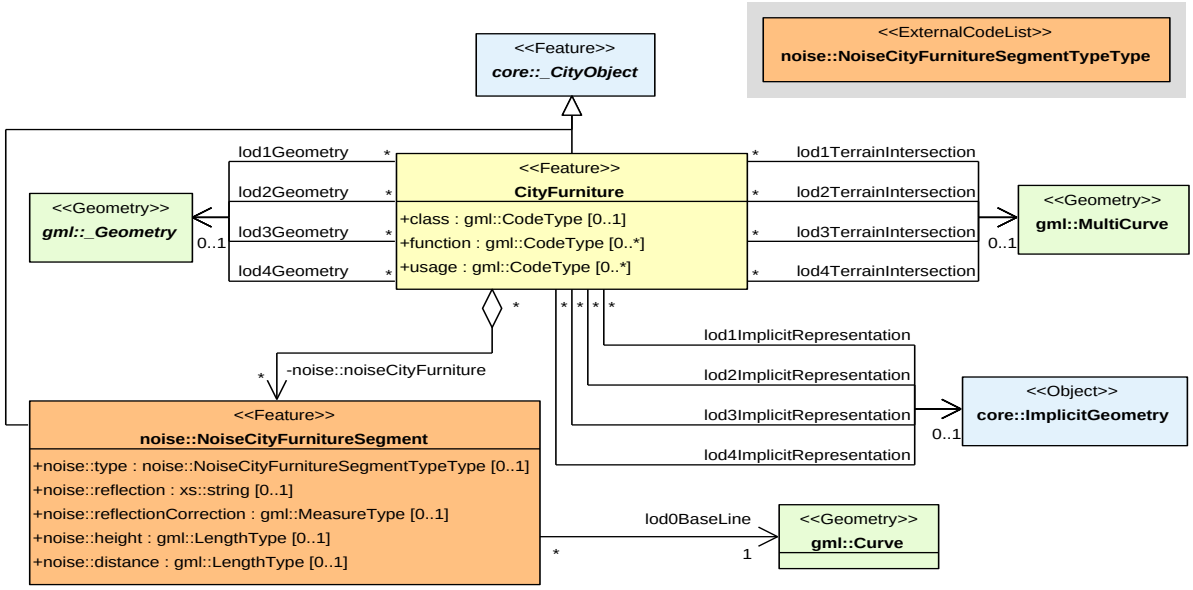

Fig. 101: CityGML noise application schema – city furniture model (light yellow=CityGML *CityFurniture* module, light orange=CityGML Noise ADE). Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *CityFurniture* module. The prefix *noise* is associated with the CityGML Noise ADE (source: Institute of Geodesy and Geoinformation Uni Bonn).

Header of the Noise ADE Schema definition file

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns="http://www.citygml.org/ade/noise_de/2.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xmlns:frn="http://www.opengis.net/citygml/cityfurniture/2.0"
  xmlns:tran="http://www.opengis.net/citygml/transportation/2.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"targetNamespace="http://www.citygml.org/ade/noise_de/2.0"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/2.0"
              schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/transportation/2.0"
              schemaLocation="http://schemas.opengis.net/citygml/transportation/2.0/transportation.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/building/2.0"
              schemaLocation="http://schemas.opengis.net/citygml/building/2.0/building.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/cityfurniture/2.0"
              schemaLocation="http://schemas.opengis.net/citygml/cityfurniture/2.0/cityFurniture.xsd"/>
  …
</xsd:schema>
```

NoiseCityFurnitureSegmentTypeType, NoiseCityFurnitureSegment

```
<xsd:element name="noiseCityFurnitureSegmentProperty" type="NoiseCityFurnitureSegmentPropertyType"
  substitutionGroup="frn:_GenericApplicationPropertyOfCityFurniture"/>
<!-- ========================================================================================== -->
<xsd:complexType name="NoiseCityFurnitureSegmentPropertyType">
  <xsd:sequence minOccurs="0">
    <xsd:element ref="NoiseCityFurnitureSegment" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xsd:complexType>
<!-- ========================================================================================== -->
<xsd:complexType name="NoiseCityFurnitureSegmentType">
  <xsd:complexContent>
    <xsd:extension base="core:AbstractCityObjectType">
      <xsd:sequence>
        <xsd:element name="type" type="gml:CodeType" minOccurs="0"/>
        <xsd:element name="reflection" type="xsd:string" minOccurs="0"/>
        <xsd:element name="reflectionCorrection" type="gml:MeasureType" minOccurs="0"/>
        <xsd:element name="height" type="gml:LengthType" minOccurs="0"/>
        <xsd:element name="distance" type="gml:LengthType" minOccurs="0"/>
        <xsd:element name="lod0BaseLine" type="gml:CurvePropertyType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<!-- ========================================================================================== -->
<xsd:element name="NoiseCityFurnitureSegment" type="NoiseCityFurnitureSegmentType" substitutionGroup="core:_CityObject"/>
```

NoiseRoadSegmentType, NoiseRoadSegment

```
<xsd:element name="noiseRoadSegmentProperty" type="NoiseRoadSegmentPropertyType"
  substitutionGroup="tran:_GenericApplicationPropertyOfRoad"/>
<!-- ========================================================================================== -->
<xsd:complexType name="NoiseRoadSegmentPropertyType">
  <xsd:sequence minOccurs="0">
    <xsd:element ref="NoiseRoadSegment"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xsd:complexType>
<!-- ========================================================================================== -->
<xsd:complexType name="NoiseRoadSegmentType">
  <xsd:complexContent>
    <xsd:extension base="tran:AbstractTransportationObjectType">
      <xsd:sequence>
        <xsd:element name="mDay" type="gml:MeasureType" minOccurs="0"/>
        <xsd:element name="mEvening" type="gml:MeasureType" minOccurs="0"/>
        <xsd:element name="mNight" type="gml:MeasureType" minOccurs="0"/>
```

```
          <xsd:element name="mDay16" type="gml:MeasureType" minOccurs="0"/>
          <xsd:element name="pDay" type="gml:MeasureType" minOccurs="0"/>
          <xsd:element name="pEvening" type="gml:MeasureType" minOccurs="0"/>
          <xsd:element name="pNight" type="gml:MeasureType" minOccurs="0"/>
          <xsd:element name="pDay16" type="gml:MeasureType" minOccurs="0"/>
          <xsd:element name="dtv" type="gml:MeasureType" minOccurs="0"/>
          <xsd:element name="speedDayPkw" type="gml:SpeedType" minOccurs="0"/>
          <xsd:element name="speedEveningPkw" type="gml:SpeedType" minOccurs="0"/>
          <xsd:element name="speedNightPkw" type="gml:SpeedType" minOccurs="0"/>
          <xsd:element name="speedDayLkw" type="gml:SpeedType" minOccurs="0"/>
          <xsd:element name="speedEveningLkw" type="gml:SpeedType" minOccurs="0"/>
          <xsd:element name="speedNightLkw" type="gml:SpeedType" minOccurs="0"/>
          <xsd:element name="roadSurfaceMaterial" type="xsd:string" minOccurs="0"/>
          <xsd:element name="roadSurfaceCorrection" type="gml:MeasureType" minOccurs="0"/>
          <xsd:element name="distanceCarriageway" type="gml:LengthType" minOccurs="0"/>
          <xsd:element name="distanceD" type="gml:LengthType" minOccurs="0"/>
          <xsd:element name="bridge" type="xsd:boolean" minOccurs="0"/>
          <xsd:element name="tunnel" type="xsd:boolean" minOccurs="0"/>
          <xsd:element name="roadGradientPercent" type="gml:MeasureType" minOccurs="0"/>
          <xsd:element name="lod0BaseLine" type="gml:CurvePropertyType"/>
          <xsd:element name="lineage" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- ========================================================================================== -->
  <xsd:element name="NoiseRoadSegment" type="NoiseRoadSegmentType" substitutionGroup="core:_CityObject"/>
```

NoiseRailwaySegmentType, NoiseRailwaySegment

```
<xsd:complexType name="NoiseRailwaySegmentPropertyType">
  <xsd:sequence minOccurs="0">
    <xsd:element ref="NoiseRailwaySegment"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xsd:complexType>
<!-- ========================================================================================== -->
<xsd:complexType name="NoiseRailwaySegmentType">
  <xsd:complexContent>
    <xsd:extension base="tran:AbstractTransportationObjectType">
      <xsd:sequence>
        <xsd:element name="railwaySurfaceMaterial" type="xsd:string" minOccurs="0"/>
        <xsd:element name="railwaySurfaceCorrection" type="gml:MeasureType" minOccurs="0"/>
        <xsd:element name="bridge" type="xsd:boolean" minOccurs="0"/>
        <xsd:element name="crossing" type="xsd:boolean" minOccurs="0"/>
        <xsd:element name="curveRadius" type="gml:LengthType" minOccurs="0"/>
        <xsd:element name="additionalCorrectionSegment" type="gml:MeasureType" minOccurs="0"/>
        <xsd:element name="lod0BaseLine" type="gml:CurvePropertyType"/>
        <xsd:element name="usedBy" type="TrainPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<!-- ========================================================================================== -->
<xsd:element name="NoiseRailwaySegment" type="NoiseRailwaySegmentType" substitutionGroup="core:_CityObject"/>
```

TrainType, TrainPropertyType

```
<xsd:complexType name="TrainPropertyType">
  <xsd:sequence>
    <xsd:element name="Train" type="TrainType"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xsd:complexType>
<!-- ========================================================================================== -->
<xsd:complexType name="TrainType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element name="trainType" type="xsd:string"/>
        <xsd:element name="trainTypeCorrection" type="gml:MeasureType" minOccurs="0"/>
        <xsd:element name="brakePortionDay" type="gml:MeasureType" minOccurs="0"/>
```

```
        <xsd:element name="brakePortionEvening" type="gml:MeasureType" minOccurs="0"/>
        <xsd:element name="brakePortionNight" type="gml:MeasureType" minOccurs="0"/>
        <xsd:element name="lengthDay" type="gml:LengthType" minOccurs="0"/>
        <xsd:element name="lengthEvening" type="gml:LengthType" minOccurs="0"/>
        <xsd:element name="lengthNight" type="gml:LengthType" minOccurs="0"/>
        <xsd:element name="speedDay" type="gml:SpeedType" minOccurs="0"/>
        <xsd:element name="speedEvening" type="gml:SpeedType" minOccurs="0"/>
        <xsd:element name="speedNight" type="gml:SpeedType" minOccurs="0"/>
        <xsd:element name="additionalCorrectionTrain" type="gml:MeasureType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Application specific attributes for _AbstractBuilding

```
  <xsd:element name="buildingReflection" type="xsd:string"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xsd:element name="buildingReflectionCorrection" type="gml:MeasureType"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xsd:element name="buildingLDenMax" type="gml:MeasureType"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xsd:element name="buildingLDenMin" type="gml:MeasureType"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xsd:element name="buildingLDenEq" type="gml:MeasureType"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xsd:element name="buildingLNightMax" type="gml:MeasureType"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xsd:element name="buildingLNightMin" type="gml:MeasureType"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xsd:element name="buildingLNightEq" type="gml:MeasureType"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xsd:element name="buildingHabitants" type="xsd:positiveInteger"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xsd:element name="buildingAppartments" type="xsd:positiveInteger"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xsd:element name="buildingImmissionPoints" type="gml:integerList"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xsd:element name="remark" type="xsd:string"
               substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
```

H.2 Example dataset

The following dataset illustrates a CityGML instance document which uses the application noise schema. It contains two CityObject features: a road object and a building object. The dataset references the XML Schema definition file of the CityGML Noise ADE which explicitly imports the XML Schema definitions of the CityGML modules extended by the Noise ADE (*CityGML Core*, *Building*, *Transportation*, and *CityFurniture* module). Thus, all classes defined by the employed CityGML modules can be used in the instance document. Furthermore, the application specific additions such as new object types (e.g. *NoiseRoadSegment*) and additional thematic attributes (e.g. the attributes defined for *_AbstractBuilding*) are available. These additional elements are distinguished from standard CityGML elements by the namespace prefix *noise* which refers to the noise schema definition.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CityModel xmlns="http://www.opengis.net/citygml/2.0"
  xmlns:tran="http://www.opengis.net/citygml/transportation/2.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xmlns:noise="http://www.citygml.org/ade/noise_de/2.0"
  xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.citygml.org/ade/noise_de NoiseADE/CityGML-NoiseADE.xsd">
  <gml:boundedBy>
    <gml:Envelope srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
      <gml:pos srsDimension="3">5616000.0 2540097.5 54.5</gml:pos>
      <gml:pos srsDimension="3">5673522.3 2576495.6 172.9</gml:pos>
    </gml:Envelope>
  </gml:boundedBy>
  <cityObjectMember>
```

```
    <tran:Road gml:id="CR_0815">
      <gml:name>B1</gml:name>
      <gml:boundedBy>
        <gml:Envelope srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
          <gml:pos srsDimension="3">5618686.0 2573988.4 158.0</gml:pos>
          <gml:pos srsDimension="3">5618705.5 2574049.8 158.2</gml:pos>
        </gml:Envelope>
      </gml:boundedBy>
      <tran:function>B1303</tran:function>
      <noise:noiseRoadSegmentProperty>
        <noise:NoiseRoadSegment gml:id="CNRS_0815">
          <gml:boundedBy>
            <gml:Envelope srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
              <gml:pos srsDimension="3">5618686.0 2573988.4 158.0</gml:pos>
              <gml:pos srsDimension="3">5618705.5 2574049.8 158.2</gml:pos>
            </gml:Envelope>
          </gml:boundedBy>
          <noise:mDay uom="kfzph">2564.123</noise:mDay>
          <noise:mEvening uom="kfzph">145.123</noise:mEvening>
          <noise:mNight uom="kfzph">1231.123</noise:mNight>
          <noise:mDay16 uom="kfzph">2010.123</noise:mDay16>
          <noise:pDay uom="percent">25.123</noise:pDay>
          <noise:pEvening uom="percent">35.123</noise:pEvening>
          <noise:pNight uom="percent">45.123</noise:pNight>
          <noise:pDay16 uom="percent">30.123</noise:pDay16>
          <noise:dtv uom="kfzp24h">20564.123</noise:dtv>
          <noise:speedDayPkw uom="kmph">130.123</noise:speedDayPkw>
          <noise:speedEveningPkw uom="kmph">100.123</noise:speedEveningPkw>
          <noise:speedNightPkw uom="kmph">50.123</noise:speedNightPkw>
          <noise:speedDayLkw uom="kmph">80.123</noise:speedDayLkw>
          <noise:speedEveningLkw uom="kmph">80.123</noise:speedEveningLkw>
          <noise:speedNightLkw uom="kmph">50.123</noise:speedNightLkw>
          <noise:roadSurfaceMaterial>Pflaster mit ebener Oberfläche</noise:roadSurfaceMaterial>
          <noise:roadSurfaceCorrection uom="dB">2.123</noise:roadSurfaceCorrection>
          <noise:distanceCarriageway uom="m">15.123</noise:distanceCarriageway>
          <noise:distanceD uom="m">10.123</noise:distanceD>
          <noise:bridge>true</noise:bridge>
          <noise:tunnel>false</noise:tunnel>
          <noise:roadGradientPercent uom="percent">5.245</noise:roadGradientPercent>
          <noise:lod0BaseLine>
            <gml:LineString srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783" srsDimension="3">
              <gml:coordinates decimal="." cs="," ts=" ">5618686.0, 2573988.4,158.200000
                5618692.5,2574008.8,158.000000 5618705.5,2574049.8,158.100000</gml:coordinates>
            </gml:LineString>
          </noise:lod0BaseLine>
          <noise:lineage>ATKIS-LVermA</noise:lineage>
        </noise:NoiseRoadSegment>
      </noise:noiseRoadSegmentProperty>
      …
    </tran:Road>
  </cityObjectMember>
  <cityObjectMember>
    <bldg:Building gml:id="UUID_ef6e19e3-c412-440b-8ba9-24900aa173b5">
      <gml:name>small building</gml:name>
      <creationDate>2007-01-04</creationDate>
      <bldg:function>1060</bldg:function>
      <bldg:measuredHeight uom="m">2.38</bldg:measuredHeight>
      <bldg:lod1Solid>
        <gml:Solid>
          <gml:exterior>
            <gml:CompositeSurface>
              <gml:surfaceMember>
                <gml:Polygon srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
                  <gml:outerBoundaryIs>
                    <gml:LinearRing>
                      <gml:coordinates cs="," decimal="." ts=" ">5662497.03,2559357.47,38.2357750703488
                        5662489.23,2559355.51,38.2357750703488 5662488.178,2559355.247,38.2357750703488
                        5662489.022,2559351.872,38.2357750703488 5662497.877,2559354.097,38.2357750703488
                        5662501.43,2559354.99,38.2357750703488 5662500.584,2559358.357,38.2357750703488
                        5662497.03,2559357.47,38.2357750703488</gml:coordinates>
                    </gml:LinearRing>
                  </gml:outerBoundaryIs>
                </gml:Polygon>
              </gml:surfaceMember>
              …
            </gml:CompositeSurface>
```

```
            </gml:exterior>
          </gml:Solid>
        </bldg:lod1Solid>
        <bldg:address>
          <Address>
            <xalAddress>
              <xAL:AddressDetails>
                <xAL:Country>
                  <xAL:CountryName>Germany</xAL:CountryName>
                  <xAL:Locality Type="Town">
                    <xAL:LocalityName>Musterstadt</xAL:LocalityName>
                    <xAL:Thoroughfare Type="Street">
                      <xAL:ThoroughfareNumber>1</xAL:ThoroughfareNumber>
                      <xAL:ThoroughfareName>Musterstrasse</xAL:ThoroughfareName>
                    </xAL:Thoroughfare>
                    <xAL:PostalCode>
                      <xAL:PostalCodeNumber>10000</xAL:PostalCodeNumber>
                    </xAL:PostalCode>
                  </xAL:Locality>
                </xAL:Country>
              </xAL:AddressDetails>
            </xalAddress>
          </Address>
        </bldg:address>
        <noise:buildingReflection>Fassade</noise:buildingReflection>
        <noise:buildingReflectionCorrection uom="dB">3.23</noise:buildingReflectionCorrection>
        <noise:buildingLDenMax uom="dB">10</noise:buildingLDenMax>
        <noise:buildingLDenMin uom="dB">30</noise:buildingLDenMin>
        <noise:buildingLDenEq uom="dB">20</noise:buildingLDenEq>
        <noise:buildingLNightMax uom="dB">40</noise:buildingLNightMax>
        <noise:buildingLNightMin uom="dB">60</noise:buildingLNightMin>
        <noise:buildingLNightEq uom="dB">50</noise:buildingLNightEq>
        <noise:buildingHabitants>32</noise:buildingHabitants>
        <noise:buildingAppartments>8</noise:buildingAppartments>
        <noise:buildingImmissionPoints>45 1 1 1 50 2 2 2 </noise:buildingImmissionPoints>
      </bldg:Building>
    </cityObjectMember>
</CityModel>
```

Listing 14: Excerpt from a CityGML dataset implementing the illustrated CityGML noise application schema.

Annex I
(informative)

Example ADE for Ubiquitous Network Robots Services

This annex illustrates the usage of CityGML within Robotics services. The definition of the corresponding *Application Domain Extension* (ADE) is included as an example.

I.1 Overview of Ubiquitous Network Robots

In the field of robotics, there is a concept which originated in Japan called *network robots*, which aims to achieve advanced robot services by connecting multiple robots. The goal is to go beyond the capacity limits of a single robot. Sensors embedded in the environment and agents on the internet are also broadly regarded as robots, and the research and development of network robot technology that connects robots through a network has been promoted. Connecting multiple robots provides improved cognitive ability and interactive capability compared to a single robot, and it is thought that advanced robot services are achievable.

In Japan, the Network Robot Forum was established in 2003. Its goal is to smoothly and effectively advance the research and development of network robots and their standardization. There are approximately 150 members including machinery manufacturers, electronic manufacturers, telecommunications carriers, research institutes, academic experts, economic organizations, and local government. They engage in activities such as research, promotion, development, and standardization of network robots.

At a national level, the Ministry of Internal Affairs and Communications has started "Research and development of ubiquitous network robot technology (UNR) for the elderly and physically challenged" since 2009, and the research and development for *ubiquitous network robots* is fully in progress in Japan.

In the research and development projects for ubiquitous network robots, the extension of network robot services from a single point to multiple points is being considered. In order to expand the activities of network robots and provide various robot services, it is necessary to have a framework that controls the information about surrounding environments as spatial attributes and provides services in a way that robots can understand. In these research and development projects, the database that controls the spatial attributes for network robots is referred to as the *spatial master database*. On the assumption that robots move around inside and outside buildings, CityGML which is able to provide models for the representation of both outdoor and indoor features has been used in the spatial master database. However, there was a lack of essential information for mobile robots, such as storey structuring, flooring materials, and door attributes. Therefore, the CityGML data model was extended using an ADE mechanism. Fig. 102 shows an example 3D visualization of a spatial master database, which is generated from CityGML with Robotics ADE and is used in these research projects.

The coordinate system used for indoor positioning of mobile robots is not a global coordinate system as used outdoors, but a local coordinate system configured for each building. If a building has multiple floors, different coordinate systems might be applied for each floor. Furthermore, multiple coordinate systems might be applied for one floor. For interaction with humans, it is important to control multiple coordinate systems set up for each building and manage the semantic knowledge of individual objects by using a spatial master database.

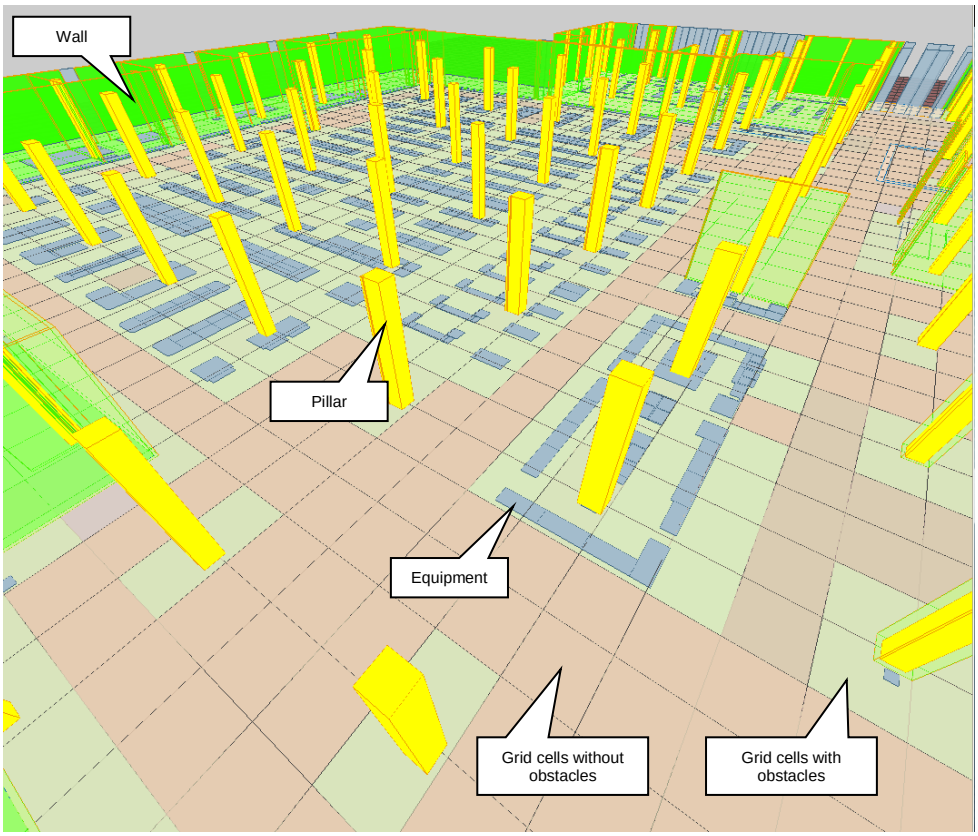

Fig. 102: 3D spatial master database for a shopping mall. This was generated from CityGML with Robotics ADE and used for the ubiquitous network robots services research project in Japan. The height of each floor is set up when turning the original 2D CAD data into 3D data in the spatial master database. Thus, the heights of some equipment (for example, furniture) is not set up and is displayed in 2D. Also, the grid map (with obstacles information) generated by the grid map function of the spatial master database is attached to the ground on the floor surface. The grid map consists of grid cells of a given size, and is color-coded depending on whether obstacles exist or not. Robots use these grid maps to recognize whether they can pass through each grid (green=walls, yellow=pillars, blue=equipment, green floor=grid cells with obstacles such as pillars and furniture, red floor=grid cells without obstacles).

In the research and development project, a demonstration experiment was conducted at a commercial facility in 2010. Fig. 103 shows the experiment conducted at a shopping mall. Robots provided services such as talking with customers, showing customers around the store, and assisting with shopping. The shopping mall facility consists of multiple buildings, and services, in which robots move around inside and outside the buildings and show customers around the store, were also experimented with.

Spatial master database information was converted into a grid map (raster map) format, and was provided to the robots by a server. The grid map is a PNG image format, and four types of grid maps are supported: obstacles, flooring materials, floor slope, and floor bumps. The grid map is the flooring surface divided into grid cells of a given size and information for each grid cell is coded. The information of each grid cell is displayed by different color pixels in an image format of grid map. The information about flooring materials, floor slope, and floor bumps is required because such items can become impassable obstacles depending on the functions and constraints of the robot-moving mechanism. In the shopping mall used for the demonstration experiment, there are some high-angle slopes in the hallway between buildings. Two different types of robots were used in the experiment, one type was able to go over the slope and the other was not. The latter robot calls the former robot through a network and turned over the customer guidance, when it arrives at an impassable zone indicated in the grid maps.

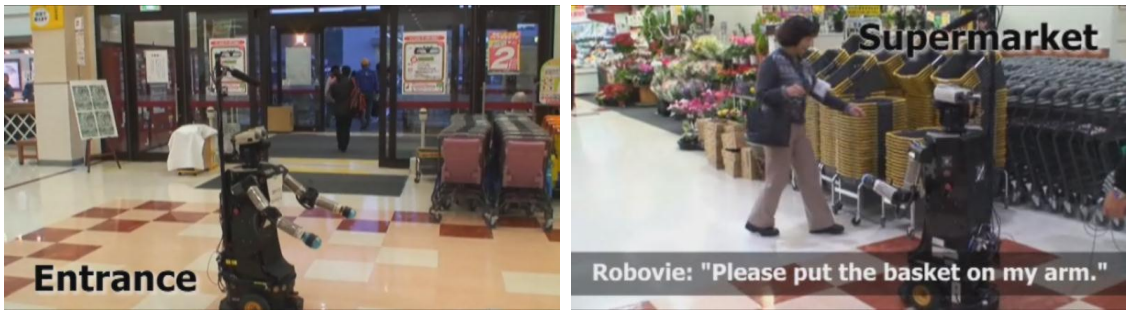

Fig. 103: The scene from the demonstration experiment conducted at a shopping mall using a Robot (Robovie-II by ATR in Japan) (source: "Robovie II helps elderly customers in a supermarket", ©ATR (Advanced Telecommunications Research Institute International).

I.2 Overview of the Spatial Master Database

This section gives an overview of the spatial master databases used for network robots.

When the data model for a spatial master database is constructed, the compatibility with CityGML LOD4 is taken into consideration. Moreover, the concepts of flooring materials (*materialType*), *Storey*, and door attributes (*doorOperationType*), which are essential for mobile robots, are added. Section I.3 provides more details.

Fig. 104 shows the configuration of a spatial master database management system used in the demonstration experiment described in section I.1. This system imports information of a building, such as CAD data or CityGML data, and stores it in spatial master database. And when converting it into a grid map, this system searches the targeted 3D objects and projects them onto a 2D floor surface. The next step is to divide it into grid, designating floor number, floor area (Boundary Box) and grid size (width and height) that the floor surface is divided into. Finally, it generates PNG images whose pixel values are related to each grid information, such as flooring materials or whether some obstacles exist or not. After converting the 3D model to this grid map, this system provides it to the Area management gateway server.

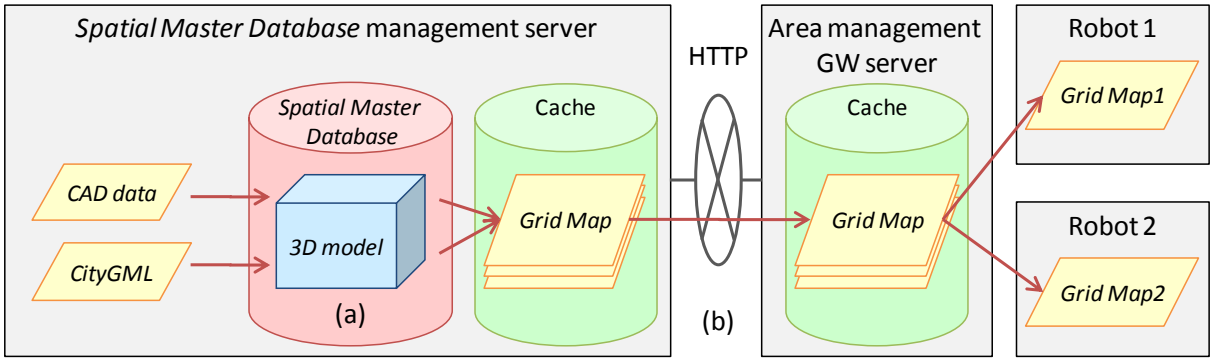

Fig. 104: Configuration of a spatial master database management system.

The shopping mall where the demonstration experiment was conducted consists of two buildings (indoor) and a hallway (outdoor) that connects the buildings (shown in Fig. 105). The spatial master database mainly stores the indoor spatial data, and stores outdoor spatial data only for the hallway. Fig. 106 shows an example of a generated grid map for which the pixel values indicate flooring materials. The service providing the grid map is developed as an extended WMS interface. This WMS interface is based on the work which was developed in "Outdoor and Indoor 3D Routing Services Engineering Report" (cf. Sato 2009, OGC Doc. No. 09-067r2).

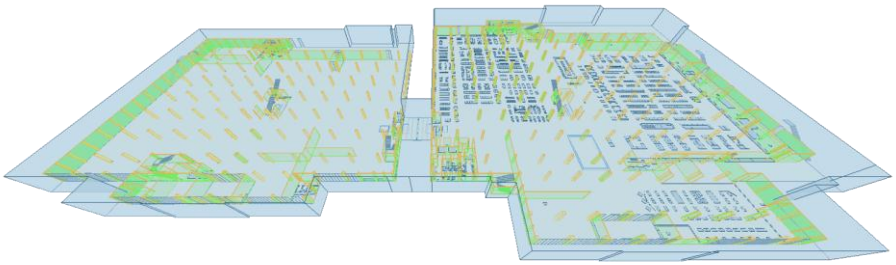

Fig. 105: 3D model of a shopping mall, which consists of two buildings (indoor) and a hallway (outdoor) that connects the buildings.

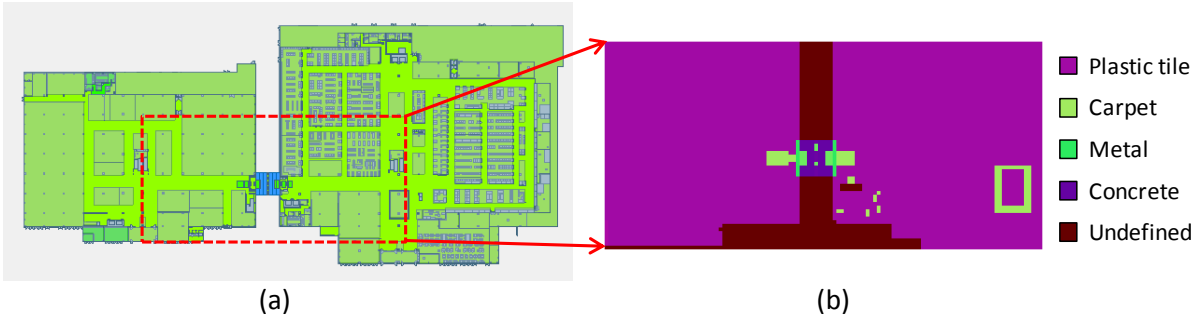

Fig. 106: Example of generating a grid map (b) from spatial data (a), when the grid map is flooring material.

I.3 Overview of the CityGML ADE

This section describes the implementation of the spatial master database explained in section I.2 by using CityGML. Specifically, this section shows UML diagrams and XML schemas for the data model of the Robotics application schema that was implemented as two independent ADEs. The reason for preparing two ADEs is that there are two types of information which is needed for Ubiquitous Network Robots service. One ADE is for more general information which can be used not only for robots service but also for indoor service. The other ADE is for information which is focused on robots service. The first ADE is the *CityGML Standard Opening ADE*, which is a schema extending the *_Opening*, *Door*, and *Window* classes of CityGML. The second ADE is the *UNR (Ubiquitous Network Robots) ADE*, which is a schema adding classes such as *Storey* and attributes such as door type and floor materials that are essential for ubiquitous network robots.

The purpose of this section is to provide an example showing how CityGML can be extended using multiple ADEs. As the semantics of the specific attributes and object types, which are implemented as codelists, result from the Japanese national research project for Ubiquitous Network Robots services, they are not explained in detail here. This section explains the Standard Opening ADE and UNR ADE. ADE details are explained in the CityGML Wiki (see http://www.citygmlwiki.org/index.php/CityGML-ADEs). The XML Schema definition of both ADEs together with example datasets and code lists can be additionally obtained from http://schemas.opengis.net/citygml/examples/2.0/ade/robotics-ade/.

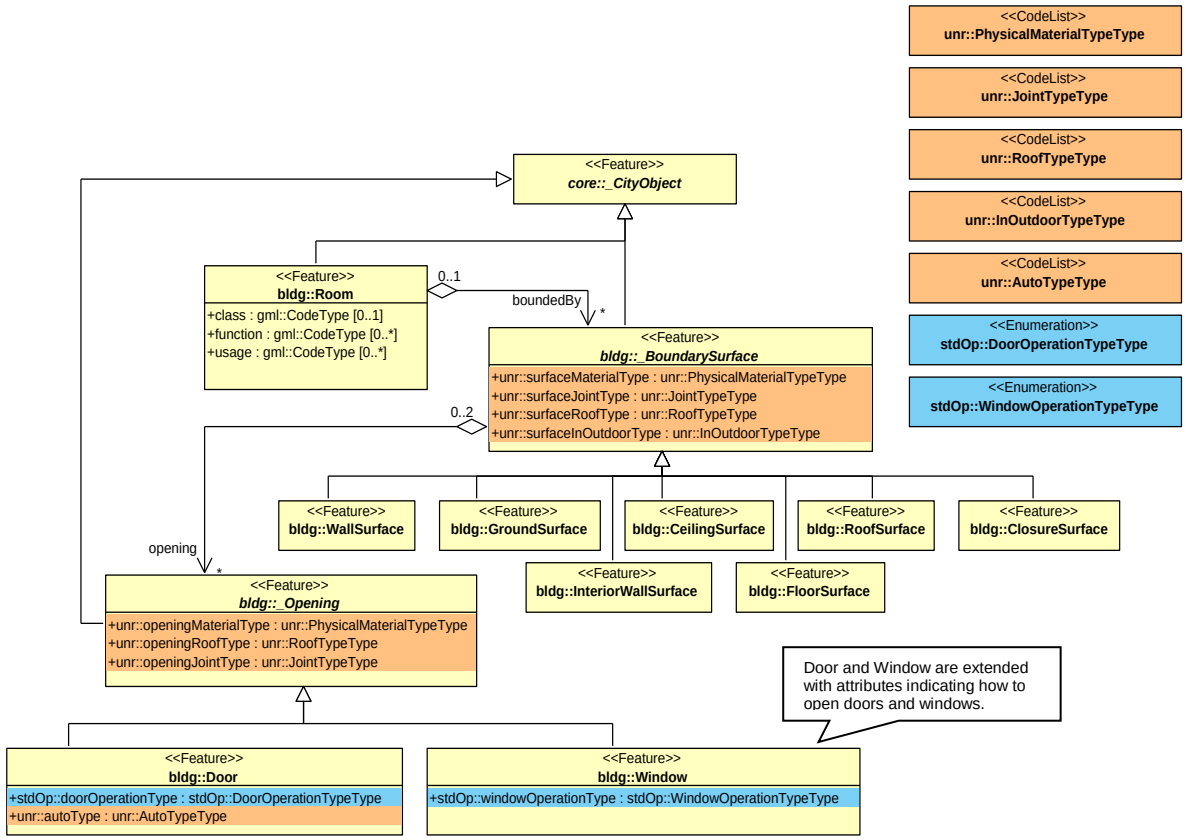

Fig. 107: CityGML Robotics application schema – Standard Openings and UNR models including Door and Window (light yellow=CityGML modules, light blue=CityGML Standard Opening ADE, light orange=UNR ADE). Prefixes are used to indicate XML namespaces associated with model elements. The prefix *stdOp* is associated with the CityGML Standard Opening ADE (source:Institute for Geodesy and Geoinformation Science, Technical University Berlin), and the prefix *unr* is associated with the UNR ADE (source: Central Research Laboratory, Hitachi, Ltd.). The CityGML model elements are associated with the recommended CityGML prefixes.

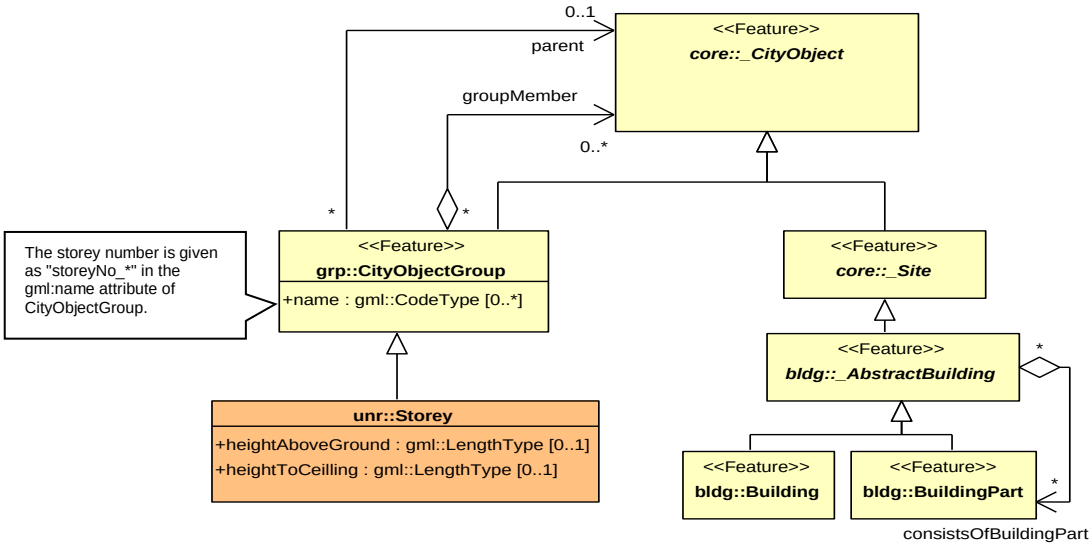

Fig. 108: CityGML Robotics application schema – UNR Storey model (light yellow=CityGML module, light orange=UNR ADE). Prefixes are used to indicate XML namespaces associated with model elements. The prefix *unr* is associated with the UNR ADE (source: Central Research Laboratory, Hitachi, Ltd.). The CityGML model elements are associated with the recommended CityGML prefixes.

Header of the Standard Openings ADE Schema definition file

```
<xsd:schema xmlns="http://unr.crl.hitachi.co.jp/ade/standard_opening"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  targetNamespace="http://unr.crl.hitachi.co.jp/ade/standard_opening" elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.opengis.net/gml"
              schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/building/2.0"
              schemaLocation="http://schemas.opengis.net/citygml/building/2.0/building.xsd"/>
  …
</xsd:schema>
```

Application specific attributes for Door

```
<xsd:element name="doorOperationType" type="DoorOperationTypeType"
  substitutionGroup="bldg:_GenericApplicationPropertyOfDoor"/>
<xsd:simpleType name="DoorOperationTypeType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="swinging"/>
    <xsd:enumeration value="double_acting"/>
    <xsd:enumeration value="sliding"/>
    <xsd:enumeration value="folding"/>
    <xsd:enumeration value="revolving"/>
    <xsd:enumeration value="rollingup"/>
    <xsd:enumeration value="userdefined"/>
    <xsd:enumeration value="notdefined"/>
  </xsd:restriction>
</xsd:simpleType>
```

Application specific attributes for Window

```
<xsd:element name="windowOperationType" type="WindowOperationTypeType"
    substitutionGroup="bldg:_GenericApplicationPropertyOfDoor"/>
<xsd:simpleType name="WindowOperationTypeType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="sidehungrighthand"/>
    <xsd:enumeration value="sidehunglefthand"/>
    <xsd:enumeration value="tiltandturnrighthand"/>
    <xsd:enumeration value="tiltandturnlefthand"/>
    <xsd:enumeration value="tophung"/>
    <xsd:enumeration value="bottomhung"/>
    <xsd:enumeration value="pivothorizontal"/>
    <xsd:enumeration value="pivotvertical"/>
    <xsd:enumeration value="slidinghorizontal"/>
    <xsd:enumeration value="slidingvertical"/>
    <xsd:enumeration value="removablecasement"/>
    <xsd:enumeration value="fixedcasement"/>
    <xsd:enumeration value="otheroperation"/>
    <xsd:enumeration value="notdefined"/>
  </xsd:restriction>
</xsd:simpleType>
```

Header of the UNR ADE Schema definition file

```
<xsd:schema xmlns="http://unr.crl.hitachi.co.jp/ade/unr"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:core="http://www.opengis.net/citygml/2.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xmlns:grp="http://www.opengis.net/citygml/cityobjectgroup/2.0"
  targetNamespace="http://unr.crl.hitachi.co.jp/ade/unr" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.opengis.net/gml"
              schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/building/2.0"
              schemaLocation="http://schemas.opengis.net/citygml/building/2.0/building.xsd"/>
```

```
  <xsd:import namespace="http://www.opengis.net/citygml/cityobjectgroup/2.0"
              schemaLocation="http://schemas.citygml.org/citygml/cityobjectgroup/2.0/cityObjectGroup.xsd"/>
  …
</xsd:schema>
```

UNR StoreyType, Storey

```
<xsd:complexType name="StoreyPropertyType">
  <xsd:sequence minOccurs="0">
    <xsd:element ref="Storey"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xsd:complexType>
<xsd:element name="storeyProperty" type="StoreyPropertyType"
  substitutionGroup="grp:_GenericApplicationPropertyOfCityObjectGroup"/>

<xsd:complexType name="StoreyType">
  <xsd:complexContent>
    <xsd:extension base="grp:CityObjectGroupType">
      <xsd:sequence>
        <xsd:element name="heightAboveGround" type="gml:LengthType" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="heightToCeiling" type="gml:LengthType" minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Storey" type="StoreyType" substitutionGroup="grp:CityObjectGroup"/>
```

Application specific attributes for _Opening

```
<xsd:element name="openingMaterialType" type="PhysicalMaterialTypeType"
  substitutionGroup="bldg:_GenericApplicationPropertyOfOpening"/>
<xsd:element name="openingRoofType" type="RoofTypeType" substitutionGroup="bldg:_GenericApplicationPropertyOfOpening"/>
<xsd:element name="openingJointType" type="JointTypeType" substitutionGroup="bldg:_GenericApplicationPropertyOfOpening"/>
```

Application specific attributes for Door

```
<xsd:element name="autoType" type="AutoTypeType" substitutionGroup="bldg:_GenericApplicationPropertyOfDoor"/>
```

Application specific attributes for _BoundarySurface

```
<xsd:element name="surfaceMaterialType" type="PhysicalMaterialTypeType"
  substitutionGroup="bldg:_GenericApplicationPropertyOfBoundarySurface"/>
<xsd:element name="surfaceRoofType" type="RoofTypeType"
  substitutionGroup="bldg:_GenericApplicationPropertyOfBoundarySurface"/>
<xsd:element name="surfaceJointType" type="JointTypeType"
  substitutionGroup="bldg:_GenericApplicationPropertyOfBoundarySurface"/>
<xsd:element name="surfaceInOutdoorType" type="InOutdoorTypeType"
  substitutionGroup="bldg:_GenericApplicationPropertyOfBoundarySurface"/>
```

I.4 Example Dataset

The following dataset illustrates a CityGML instance document that uses the Robotics application schema. It contains three _CityObject features: a *CityObjectGroup* object with a *Storey* object property, a *FloorSurface* object with a material type attribute, and a *Door* object with a *doorOperationType* attribute. The dataset references the XML schema definition files of the Robotics ADE, which explicitly imports the XML schema definitions of the CityGML modules extended by the CityGML Standard Opening ADE (*CityGML Core* and *Building* module) and the UNR ADE (*CityGML Core*, *Building*, and *CityObjectGroup* module). Thus, all classes defined by the employed CityGML modules can be used in the instance document. Furthermore, the application specific additions such as new object types (e.g. *Storey*) and additional thematic attributes (e.g. the attributes defined for

Door) are available. These additional elements are distinguished from standard CityGML elements by the namespace prefix *stdOp* and *unr* which refer to the Robotics schema definition.

In order to use multiple ADE files, it is necessary to follow the XML schema rules and explicitly specify in the *xsi:schemaLocation* tag that multiple ADE files (of the XML schema) are to be read. In the following dataset, the two pairs of schema location, Standard Opening ADE and UNR ADE, are coded in *xsi:schemaLocation*. When software such as a CityGML parser is implemented, multiple XML schema files must be read. Also, when multiple ADE files are used, problems such as circular references might occur and should be avoided carefully.

```
<?xml version="1.0" encoding="UTF-8"?>
<CityModel xmlns="http://www.opengis.net/citygml/2.0" xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xmlns:grp="http://www.opengis.net/citygml/cityobjectgroup/2.0"
  xmlns:stdOp="http://unr.crl.hitachi.co.jp/ade/standard_opening"
  xmlns:unr="http://unr.crl.hitachi.co.jp/ade/unr"
  xsi:schemaLocation="http://unr.crl.hitachi.co.jp/ade/standard_opening http://unr.crl.hitachi.co.jp/ade/standard_opening/stdOp.xsd
                      http://unr.crl.hitachi.co.jp/ade/unr http://unr.crl.hitachi.co.jp/ade/unr/unr.xsd">

  <cityObjectMember>
    <grp:CityObjectGroup gml:id="Storey_0">
      <gml:name>Sample Storey 0</gml:name>
      <gml:name>storeyNo_0</gml:name>
      <grp:class>building separation</grp:class>
      <grp:function>lod4Storey</grp:function>
      <grp:groupMember xlink:href="#FloorSurface_1"/>
      <grp:groupMember xlink:href="#Door_1"/>
      <unr:storeyProperty>
        <unr:Storey>
          <unr:heightAboveGround uom="#m">0.0</unr:heightAboveGround>
          <unr:heightToCeiling uom="#m">5.0</unr:heightToCeiling>
        </unr:Storey>
      </unr:storeyProperty>
    </grp:CityObjectGroup>
  </cityObjectMember>

  <cityObjectMember>
    <bldg:Building>
      …
      <bldg:boundedBy>
        <bldg:FloorSurface gml:id="FloorSurface_1">
          <gml:name>Sample FloorSurface 1</gml:name>
          <bldg:lod4MultiSurface>
            <gml:MultiSurface>
              <gml:surfaceMember>
                <gml:Polygon>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:posList srsDimension="3"> 0.0 0.0 0.0 50.0 0.0 0.0 50.0 50.0 0.0 0.0 50.0 0.0 0.0 0.0 0.0
                      </gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:MultiSurface>
          </bldg:lod4MultiSurface>
          <unr:surfaceMaterialType>2491</unr:surfaceMaterialType>
          <unr:surfaceRoofType>2</unr:surfaceRoofType>
          <unr:surfaceInOutdoorType>2</unr:surfaceInOutdoorType>
          <unr:surfaceJointType>4</unr:surfaceJointType>
        </bldg:FloorSurface>
      </bldg:boundedBy>
      …
      <bldg:boundedBy>
        <bldg:WallSurface>
          …
          <bldg:opening>
            <bldg:Door gml:id="Door_1">
              <gml:name>Sample Door 1</gml:name>
              <bldg:lod4MultiSurface>
                <gml:MultiSurface>
                  <gml:surfaceMember>
                    <gml:Polygon>
```

```
                    <gml:exterior>
                      <gml:LinearRing>
                        <gml:posList srsDimension="3"> 0.0 0.0 0.0 0.0 0.0 20.0 0.0 10.0 20.0 0.0 10.0 0.0 0.0 0.0 0.0 </gml:posList>
                      </gml:LinearRing>
                    </gml:exterior>
                  </gml:Polygon>
                </gml:surfaceMember>
              </gml:MultiSurface>
            </bldg:lod4MultiSurface>
            <unr:openingMaterialType>2491</unr:openingMaterialType>
            <unr:openingRoofType>2</unr:openingRoofType>
            <unr:openingJointType>1</unr:openingJointType>
            <stdOp:doorOperationType>swinging</stdOp:doorOperationType>
          </bldg:Door>
        </bldg:opening>
      </bldg:WallSurface>
    </bldg:boundedBy>
    …
  </bldg:Building>
</cityObjectMember>

</CityModel>
```

Listing 15: Excerpt from a CityGML dataset implementing the illustrated CityGML Robotics application schema. Refer to code lists included in the schema for information about attribute values such as 2491 of *unr:openingMaterialType*.

Annex K
(informative)

Revision history

Date	Release	Editor	Description
2006-02-01	0.1.0	Czerwinski, Kolbe, Gröger	Initialisation of the document.
2006-02-22	0.1.0	Gröger, Gruber	Additions to UML diagrams.
2006-04-26	0.2.0	Kolbe, Gröger, Czerwinski	Release of CityGML draft standard to EuroSDR.
2006-03-06	0.3.0	Kolbe, Gröger, Czerwinski	Changes on property names, some attributes added. Release of CityGML standard document to OGC.
2007-05-30	0.4.0	Kolbe, Gröger, Nagel, Lorenz, Benner, Czerwinski, Gruber, Schlüter, Bildstein, Drees, Löwner	Introduction of a new appearance model. Introduction of Application Domain Extensions (ADE). Minor changes to the building model. Minor changes to city object groups. The concept of TerrainIntersectionCurves (TIC) added to CityFurniture. Adapation of code lists. Release of CityGML standard document to OGC.
2008-02-04	1.0.0	Kolbe, Gröger, Nagel, Stadler, Lorenz	Introduction of modularisation of the CityGML data model. Minor changes to the appearance model. Minor changes to city object groups and transportation objects. Encoding of external code lists changed to GML 3.1.1 Simple Dictionary Profile. Revision of UML diagrams. Release of draft CityGML standard document to CityGML 1.0 SWG.
2008-05-19	1.0.0	Kolbe, Gröger, Nagel, Stadler, Lorenz	Incorporation of changes and editorial corrections based on comments and recommendations during the OGC RFC public comment phase and corresponding actions agreed on by the CityGML 1.0 SWG. Final release of draft CityGML standard document to CityGML 1.0 SWG.
2008-8-18	1.0.0	Carl Reed	Prepare for posting as OGC™ standard and make final edits.
2011-7-19	1.1.0	Nagel, Häfele, Benner, Gröger, Gruber, Schlüter	First release of draft CityGML standard document to CityGML SWG.
2011-11-21	1.1.0	Nagel, Häfele, Gröger, Kolbe	Incorporation of changes and editorial corrections based on comments and recommendations during the OGC RFC public comment phase and corresponding actions agreed on by the CityGML SWG. Release of draft CityGML standard document to CityGML SWG.
2012-02-27	2.0.0	Nagel, Häfele, Gröger, Kolbe	Version changed from 1.1.0 to 2.0.0 in order to conform to OGC versioning policy as defined in 135r11. Reference number of this document changed to 12-019 to reflect the major version number change. Release of draft CityGML standard document to CityGML SWG.
2012-03-14	2.0.0	Carl Reed	Make final edits for publication as an OGC standard

Bibliography

Albert, J., Bachmann, M., Hellmeier, A (2003): Zielgruppen und Anwendungen für Digitale Stadtmodelle und Digitale Geländemodelle. Erhebungen im Rahmen der SIG 3D der GDI NRW (in German only). Available at http://www.ikg.uni-bonn.de/fileadmin/sig3d/pdf/Tabelle_Anwendungen_Zielgruppen.pdf.

Becker, T., Nagel, C., Kolbe, T.H. (2011): Integrated 3D modeling of multi-utility networks and their interdependencies for critical infrastructure analysis. In: Kolbe, T.H., König, G., Nagel, C. (Eds.), Advances in 3D Geo-Information Sciences. Springer, Berlin, Heidelberg, pp. 1–20.

Benner, J., Geiger, A., Leinemann, K. (2005): Flexible generation of semantic 3D building models. In: Gröger, G., Kolbe, T. H. (eds.): First International ISPRS/EuroSDR/DGPF-Workshop on Next Generation 3D City Models. Bonn, Germany, EuroSDR Publication No. 49. pp.17-22.

Berlo, L. van, Laat, R. de, (2011): Integration of BIM and GIS: The development of the CityGML GeoBIM extension. In: Kolbe, T.H., König, G., Nagel, C. (Eds.), Advances in 3D Geo-Information Sciences. Springer, Berlin, Heidelberg, pp. 211–225.

BImSchV (2006): Federal Government of Germany (eds.): 34. Verordnung zur Durchführung des Bundesimmissionsschutzgesetzes - Verordnung über die Lärmkartierung [34. BImSchV], in der Fassung vom 6.3.06, in Kraft seit 16.3.06, In: BGBl. I 06, Nr. 12, S. 516 (in German only).

Bleifuß, R., Donaubauer, A., Liebscher, J., Seitle, M., (2009): Entwicklung einer CityGML-Erweiterung für das Facility Management am Beispiel Landeshauptstadt München. In: Symposium angewandte Geoinformatik (AGIT), Salzburg.

Booch, G., Rumbaugh, J., Jacobson, I. (1997): Unified Modeling Language User Guide. Addison-Wesley.

CityGML 2007. Official homepage of CityGML. http://www.citygml.org.

Cox, S., Daisy, P., Lake, R., Portele, C., Whiteside, A. (2004): OpenGIS Geography Markup Language (GML 3.1), Implementation Specification Version 3.1.0, Recommendation Paper, OGC Doc. No. 03-105r1.

Czerwinski, A., Kolbe, T. H., Plümer, L., Stöcker-Meier, E. (2006): Interoperability and accuracy requirements for EU environmental noise mapping, In: Kremers, Horst (ed.): Proceedings, InterCarto – InterGIS 12. Berlin 2006.

Czerwinski, A., Kolbe, T. H., Plümer, L., Stöcker-Meier, E. (2006b): Spatial data infrastructure techniques for flexible noise mapping strategies, In: Tochtermann, K. / Scharl, A. (eds.): Proceedings of the 20th International Conference on Environmental Informatics - Managing Environmental Knowledge. Graz 2006.

Czerwinski, A., Sandmann, S., Stöcker-Meier, E., Plümer, L. (2007): Sustainable SDI for EU noise mapping in NRW – best practice for INSPIRE, In: International Journal for Spatial Data Infrastructure Research (IJSDIR, http://ijsdir.jrc.it), 2007, vol. 1.

Döllner, J., Kolbe, T. H., Liecke, F., Sgouros, T., Teichmann, K. (2006): The Virtual 3D City Model of Berlin - Managing, Integrating and Communicating Complex Urban Information. In: Proc. of the 25th Intern. Symposium on Urban Data Management UDMS 2006 in Aal-borg, Denmark, 15-17 May 2006.

European Union (eds.): Directive 2002/49/EG of the European Parliament and of the Council of 25 June 2002 relating to the assessment and management of environmental noise, in: Official Journal of the European Communities from 18.7.02.

Federal Government of Germany (eds.): Gesetz zur Umsetzung der EG-Richtlinie über die Bewertung und Bekämpfung von Umgebungslärm vom 24. Juni 2005, in: Bundesgesetzblatt Jg. 2005, Teil 1, Nr. 38, Bonn 29.6.05 (in German only). Available at http://217.160.60.235/BGBL/bgbl1f/bgbl105s1794.pdf.

Foley, J., van Dam, A,. Feiner, S., Hughes, J. (1995): Computer Graphics: Principles and Practice. Addison Wesley, 2nd Ed.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley.

Gröger, G., Plümer, L. (2005): How to get 3-D for the Price of 2-D – Topology and Consistency of 3-D Urban GIS. *Geoinformatica*, 9(2).

Gröger, G., Benner, J., Dörschlag, D., Drees, R., Gruber U., Leinemann K., Löwner, M.-O.(2005): Das interoperable 3D-Stadtmodell der SIG 3D, in: Zeitschrift für Vermessungswesen (in German only).

Herring, J. (2001): The OpenGIS Abstract Specification, Topic 1: Feature Geometry (ISO 19107 Spatial Schema), Version 5. OGC Document Number 01-101.

IAI (2006): International Alliance for Interoperability: IFC 2x3 – Industry Foundation Classes. http://www.iai-international.org/Model/IFC(ifcXML)Specs.html, 2006.

Khronos Group Inc., Sony Computer Inc. (June 2006): COLLADA – Digital Asset Schema Release 1.4.1 – Specification. http://www.khronos.org/files/collada_spec_1_4.pdf.

Kohlhaas, A., Mitchell, J. (2007): Modelling cities. In: Constructing the future: nD modelling, p. 372-392, Taylor & Francis

Kolbe, T. H., Gröger, G. (2003): Towards unified 3D city models. In: Schiewe, J., Hahn, M., Madden, M., Sester, M. (eds): Challenges in Geospatial Analysis, Integration and Visualization II. Proc. of Joint ISPRS Workshop, Stuttgart.

Kolbe, T. H., Gröger, G., Plümer, L. (2005): CityGML – Interoperable Access to 3D City Models, In: van Oosterom, Peter, Zlatanova, Sisi, Fendel, E.M. (eds.): Geo-information for Disaster Management. Proc. of the 1st International Symposium on Geo-information for Disaster Management, Delft, The Netherlands, March 21-23. Delft.

Kolbe, T. H., Gröger, G., Plümer, L. (2008): CityGML - 3D City Models for Emergency Response, In: Zlatanova, Li (eds.): Geospatial Information Technology for Emergency Response, ISPRS book series, Taylor & Francis.

Kolbe, T. H., Nagel, C., Lorenz, A. (2009): CityGML – OGC Standard for Photogrammetry?, In: Fritsch, D. (ed.): Photogrammetric Week, Wichmann, Heidelberg, pp. 265-277.

OASIS 2003: xNAL Name and Address Standard. Organization for the Advancement of Structured Information Standards. http://xml.coverpages.org/xnal.html

Okabe, A., Boots, B., Sugihara, K. (1992): Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. John Wiley & Sons.

Sato, A. (ed.) (2009): OWS-6 Outdoor and Indoor 3D Routing Services Engineering Report, Public Engineering Report, OGC Doc. No. 09-067r2.

Schulte, C., Coors, V., (2008): Development of a CityGML ADE for dynamic 3D flood information,. In: Joint ISCRAM-CHINA and GI4DM Conference on Information Systems on Information Systems for Crisis Management, Harbin, China, 2008.

Stadler, A., Kolbe, T. H. (2007): Spatio-Semantic Coherence in the Integration of 3D City Models. In: Proceedings of 5th International ISPRS Symposium on Spatial Data Quality ISSDQ 2007 in Enschede, The Netherlands, 13-15 June 2007

VBUS (2006): Federal Government of Germany (eds.): Vorläufige Berechnungsmethode für den Umgebungslärm an Straßen vom 15.5.2006 (in German only).

VBUSch (2006): Federal Government of Germany (eds.): Vorläufige Berechnungsmethode für den Umgebungslärm an Schienenwegen vom 10.5.2006 (in German only).

VRML97, Information technology - Computer graphics and image processing - The Virtual Reality Modeling Language (VRML) – Part 1: Functional specification and UTF-8 encoding. Part 1 of ISO/IEC Standard 14772-1:1997.

Web 3D, Information technology - Computer graphics and image processing - Extensible 3D (X3D). ISO/IEC 19775:2004. http://www.web3d.org/x3d/specifications/, 2004.

Whiteside, A. (ed.) (2009): Definition identifier URNs in OGC namespace, Version: 1.3, OGC® Best Practices, OGC Doc. No. 07-092r3.

Whiteside, A. (ed.) (2005): GML 3.1.1 Simple Dictionary Profile, Version: 1.0.0, OpenGIS® Implementation Specification, OGC Doc. No. 05-099r2.

