Open Geospatial Consortium Inc.

Date: 2008-08-20

Reference number of this OGC® project document: OGC 08-007r1

Version: 1.0.0

Category: OpenGIS® Encoding Standard

Editors: Gerhard Gröger, Thomas H. Kolbe, Angela Czerwinski, Claus Nagel

OpenGIS® City Geography Markup Language (CityGML) Encoding Standard

Document type:	OpenGIS® Standard
Document subtype:	Encoding
Document stage:	Final
Document language:	English

(Page intentionally left blank)

Contents

i. 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

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

Version 1.0 of this standards document was prepared by the CityGML 1.0 Standards Working Group (SWG) of the OGC. Future discussion and development will be lead by the 3DIM Working Group.

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

CityGML continues to be developed by the members of the Special Interest Group 3D of the initiative Geodata Infrastructure North-Rhine Westphalia (GDI NRW).

For further information see http://www.gdi-nrw.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

ii. Submitting organizations

This Implementation 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

CityGML was originally developed by the Special Interest Group 3D (SIG 3D), 2002 – 2008 - www.citygml.org.

iii. Submission contact points

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

Name	Institution	Email
Prof. Dr. Thomas H. Kolbe Claus Nagel Alexandra Stadler	Institute for Geodesy and Geoinformation Science, Technical University Berlin	kolbe@igg.tu-berlin.de nagel@igg.tu-berlin.de stadler@igg.tu-berlin.de

Dr. Gerhard Gröger Prof. Dr. Lutz Plümer Angela Czerwinski Dirk Dörschlag	Institute for Geodesy and Geoinformation, University of Bonn	groeger@ikg.uni-bonn.de pluemer@ikg.uni-bonn.de czerwinski@ikg.uni-bonn.de doerschlag@ikg.uni-bonn.de
Jim Farley	Autodesk, Inc.	jim.farley@autodesk.com
Alain Lapierre Frank Steggink Paul Scarponcini	Bentley Systems, Inc.	alain.lapierre@bentley.com frank.steggink@bentley.com paul.scarponcini@bentley.com
Carsten Rönsdorf Dave Capstick Mark Pendlington	Ordnance Survey, Great Britain	Carsten.Roensdorf@ordnancesurvey.co.uk Dave.Capstick@ordnancesurvey.co.uk Mark.Pendlington@ordnancesurvey.co.uk
Prof. Dr. Jürgen Döllner Haik Lorenz	Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam	juergen.doellner@hpi.uni-potsdam.de haik.lorenz@hpi.uni-potsdam.de

iv. Participants in development

Name	Institution	Email
Ulrich Gruber, Birgit Joemann Sandra Schlüter	District Administration Recklinghausen, Cadastre Department	ulrich.gruber@kreis-recklinghausen.de birgit.joemann2@kreis-re.de sandra.schlueter@kreis-re.de
Dr. Joachim Benner Karl-Heinz Häfele Dr. Klaus Leinemann	Institute for Applied Computer Science, Helmholtz Research Center Karlsruhe	Joachim.Benner@iai.fzk.de Karl-Heinz.Haefele@iai.fzk.de
Frank Bildstein	Rheinmetall Defence Electronics	bildstein.f@rheinmetall-de.com
Rüdiger Drees	T-Systems Enterprise Services GmbH, Bonn, Germany	Ruediger.Drees@t-systems.com
Andreas Kohlhaas	GIStec GmbH (formerly)	AKohlhaas@t-online.de
Frank Thiemann	Institute for Cartography and Geoinformatics, University of Hannover	Frank.Thiemann@ikg.uni-hannover.de
Martin Degen	City of Dortmund Cadastre Department	mdegen@stadtdo.de
Heinrich Geerling	Architekturbüro Geerling	Heinrich@geerling.de
Dr. Frank Knospe	City of Essen Cadastre and Mapping Department	frank.knospe@amt62.essen.de
Hardo Müller	Snowflake Software Ltd., UK	Hardo.Mueller@snowflakesoftware.co.uk
Martin Rechner	rechner logistik	mail@rec-log.de
Jörg Haist Daniel Holweg	Fraunhofer Institute for Computer Graphics (IGD), Darmstadt	joerg.haist@igd.fraunhofer.de daniel.holweg@igd.fraunhofer.de
Prof. Dr. Peter A. Henning	Faculty for Computer Science, University of Applied Sciences, Karlsruhe	p.henning@fh-karlsruhe.de
Rolf Wegener Stephan Heitmann	State Cadastre and Mapping Agency of North-Rhine Westphalia	wegener@lverma.nrw.de heitmann@lverma.nrw.de
Prof. Dr. Marc-O. Löwner	Institute for Geodesy and Photogrammetry, Technical University of Braunschweig	m-o.loewner@tu-bs.de

v. 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,	Changes on property names, some attributes added. Release of

		Czerwinski	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 external 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.

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 1.0 Standards Working Group and the 3D Information Management (3DIM) Working Group of the OGC: Tim Case, Paul Cote, Jeffrey Bell, Chris Body, Greg Buehler, David Burggraf, François Golay, John Herring, Jury Konga, Kai-Uwe Krause, Gavin Park, Richard Pearsall, George Percivall, Clemens Portele, Mauro Salvemini, Scott Simmons, Alessandro Triglia, David Wesloh, Tim Wilson, and Greg Yetman.

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 Inc. 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 from the previous CityGML version 0.4.0 (OGC document no. 07-062):

- Modularisation of the CityGML data model;
- Minor changes to the appearance model;
- Minor changes to city object groups;
- Minor changes to transportation objects; and
- Encoding of external code lists changed to GML 3.1.1 Simple Dictionary Profile.

With CityGML version 1.0.0, modularisation of the CityGML data model was introduced. The overall CityGML data model is thematically decomposed into a CityGML core module and extension modules. Each module is defined within its own globally unique XML namespace. Due to this modularisation approach, valid CityGML 0.4.0 instance documents are not valid CityGML 1.0.0 instance documents.

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) of the initiative Geodata Infrastructure North Rhine-Westphalia (GDI NRW) in Germany. 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 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).

A subset of CityGML has been successfully implemented and evaluated 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. Today, the official 3D city model of Berlin is based on the CityGML data model and employs CityGML as the exchange format between database, editor, and presentation systems. Also the 3D city models of Stuttgart, Bochum, Essen, Dortmund, Cologne, and Bonn are based on the CityGML model.

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).

OpenGIS® City Geography Markup Language (CityGML) Encoding Standard

1 Scope

This document is an OpenGIS® 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).

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 in the future)
    - 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

Conformace 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 08-007r1. 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 08-007r1 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 (see chapter 8.2). The appearance model (see 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
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 NRW	Geodata Infrastructure North-Rhine Westphalia
GML	Geography Markup Language
IAI	International Alliance for Interoperability
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 NRW
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).

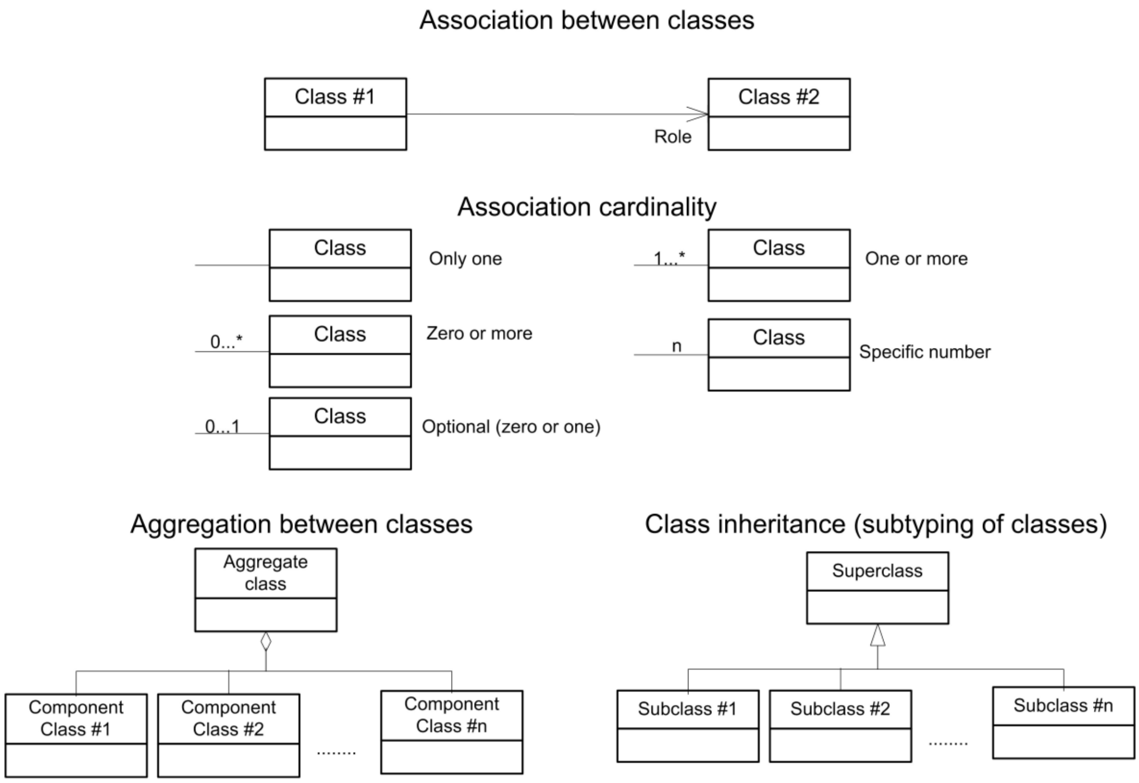


Fig. 1: UML notation (see ISO TS 19103, Geographic information - Conceptual schema language).

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*.

<<CodeList>> enumerates the valid attribute values.

<<ExternalCodeList>> enumerates the valid attributes values. In contrast to CodeList, the values are not given inline the schema but are provided within an external dictionary file. External code lists are encoded using GML 3.1.1 Simple Dictionary Profile (see chapter 6.6).

<<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”.

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/1.0	core
Appearance	http://www.opengis.net/citygml/appearance/1.0	app
Building	http://www.opengis.net/citygml/building/1.0	bldg
CityFurniture	http://www.opengis.net/citygml/cityfurniture/1.0	frn
CityObjectGroup	http://www.opengis.net/citygml/cityobjectgroup/1.0	grp
Generics	http://www.opengis.net/citygml/generics/1.0	gen
LandUse	http://www.opengis.net/citygml/landuse/1.0	luse
Relief	http://www.opengis.net/citygml/relief/1.0	dem
Transportation	http://www.opengis.net/citygml/transportation/1.0	tran
Vegetation	http://www.opengis.net/citygml/vegetation/1.0	veg
WaterBody	http://www.opengis.net/citygml/waterbody/1.0	wtr
TexturedSurface [deprecated]	http://www.opengis.net/citygml/texturedsurface/1.0	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; future extensions of CityGML will also include explicit models for bridges and tunnels), vegetation (solitary objects and also areal and volumetric biotopes), 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). Enumerative object attributes are restricted to external code lists and values 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 eleven thematic extension modules: *Appearance*, *Building*, *CityFurniture*, *CityObjectGroup*, *Generics*, *LandUse*, *Relief*, *Transportation*, *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. 2). 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 LOD 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. LOD1 is the well-known blocks model comprising prismatic buildings with flat roofs. In contrast, a building in LOD2 has differentiated roof structures and thematically differentiated surfaces. Vegetation objects may also be represented. LOD3 denotes architectural models with detailed wall and roof structures, balconies, bays and projections. High-resolution textures can be mapped onto these structures. In addition, detailed vegetation and transportation objects are components of a LOD3 model. LOD4 completes a LOD3 model by adding interior structures for 3D objects. For example, buildings are composed of rooms, interior doors, stairs, and furniture.

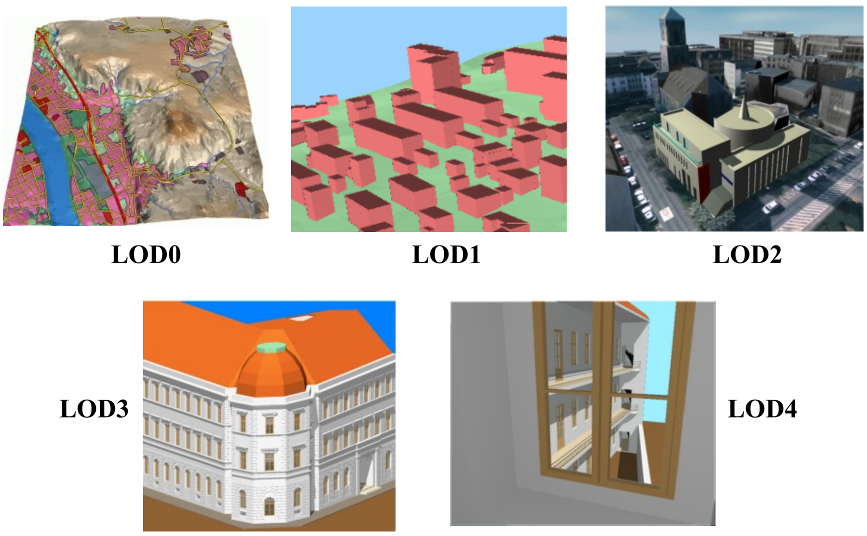


Fig. 2: The five levels of detail (LOD) defined by CityGML (source: IGG Uni Bonn)

LODs are also characterised by differing accuracies and minimal dimensions of objects (Tab. 3). The accuracy requirements given in this standard are debatable and should 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 must be 5m or less, while all objects with a footprint of at least 6m by 6m have to be considered. The positional and height accuracy of LOD2 must be 2m or better. In this LOD, all objects with a footprint of at least 4m × 4m have to be considered. Both types of accuracies in LOD3 are 0.5m, and the minimal footprint is 2m × 2m. Finally, the positional and height accuracy of LOD4 must 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 districts, projects	architectural models (outside), landmark	architectural models (interior)
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 (classification of land use)	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	-	-	-	representative exterior effects	real object form
Roof form/structure	no	flat	roof type and orientation	real object form	real object form
Roof overhanging parts	-	-	n.a.	n.a.	Yes
CityFurniture	-	important objects	prototypes	real object form	real object form
SolitaryVegetationObject	-	important objects	prototypes, higher 6m	prototypes, higher 2m	prototypes, real object form
PlantCover	-	>50*50m	>5*5m	< LOD2	<LOD2
… to be continued for the other feature themes					

Tab. 3: LOD 0-4 of CityGML with its accuracy requirements (source: 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 any *CityObjects* (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 *ClosureSurfaces. Closure-*

Surfaces 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 *ClosureSurfaces* 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. 3). However, in close-range visualisations the entrance must be treated as open. Thus, *ClosureSurfaces* are an adequate way to model those entrances.

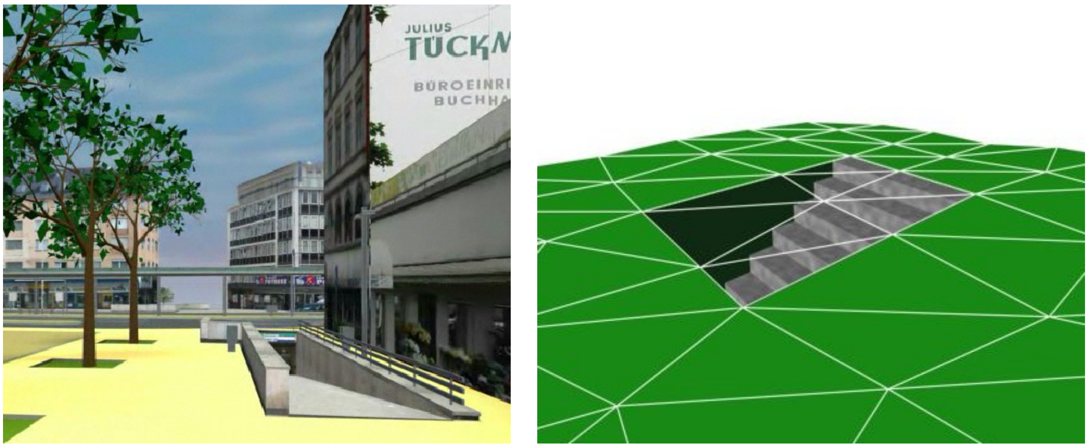


Fig. 3: Closure surfaces to seal open structures. Passages are subsurface objects (left). The entrance is sealed by a virtual *ClosureSurface,* which is both part of the DTM and the subsurface object (right) (graphic: IGG Uni Bonn).

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. 4). TICs can be applied to buildings and building parts (cf. chapter 10.3), city furniture objects (cf. chapter 10.7), and generic city objects (cf. chapter 10.10). 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.

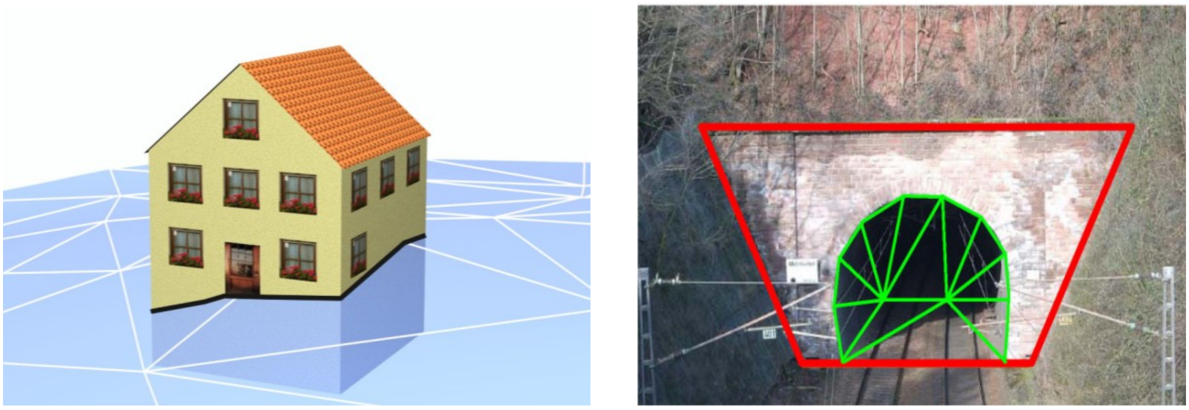


Fig. 4: *TerrainIntersectionCurve* for a building (left, black) and a tunnel object (right, white). The tunnel's hollow space is sealed by a triangulated *ClosureSurface* (graphic: IGG Uni Bonn).

6.6 *Dictionaries and external code lists for enumerative attributes*

Attributes, which are used to classify objects, often have values that are restricted to a number of discrete values. An example is the attribute *roof type*, 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. In CityGML such classifying of attributes is specified as *External-CodeLists* and implemented by *simple dictionaries* defined in the *GML 3.1.1 Simple Dictionary Profile* (cf. Whiteside 2005). Such a structure enumerates all possible values of the attribute in an external file, ensuring that the same name is used for the same notion. In addition, the translation of attribute values into other languages is facilitated.

Simple dictionaries and external code lists may be extended or redefined by users. They can have references to existing models. For example, room codes defined by the Open Standards Consortium for Real Estate (OSCRE) can be referenced instead of CityGML's predefined values. Likewise, classifications of buildings and building parts introduced by the National Building Information Model Standard (NBIMS) can be used alternatively.

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. 5). 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 have *External References* to corresponding objects in external data sets (for the UML diagram see Fig. 20; and for XML schema definition see annex A.1). 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.).

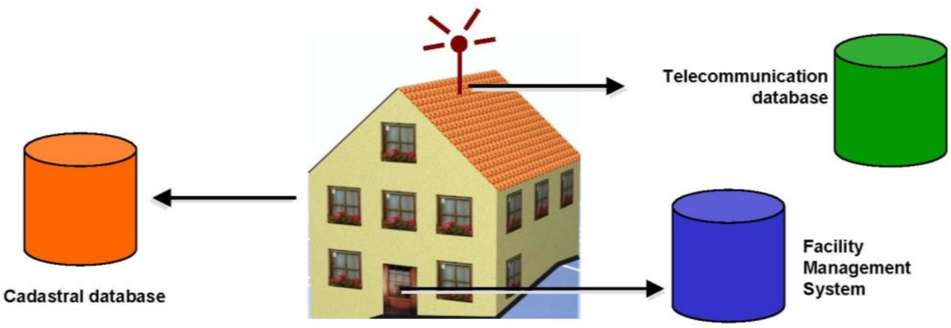


Fig. 5: External references (graphic: IGG Uni Bonn).

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 10.9; XML schema definition see annex A.5). 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 10.9).

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. 6). 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 see chapter 8.2).

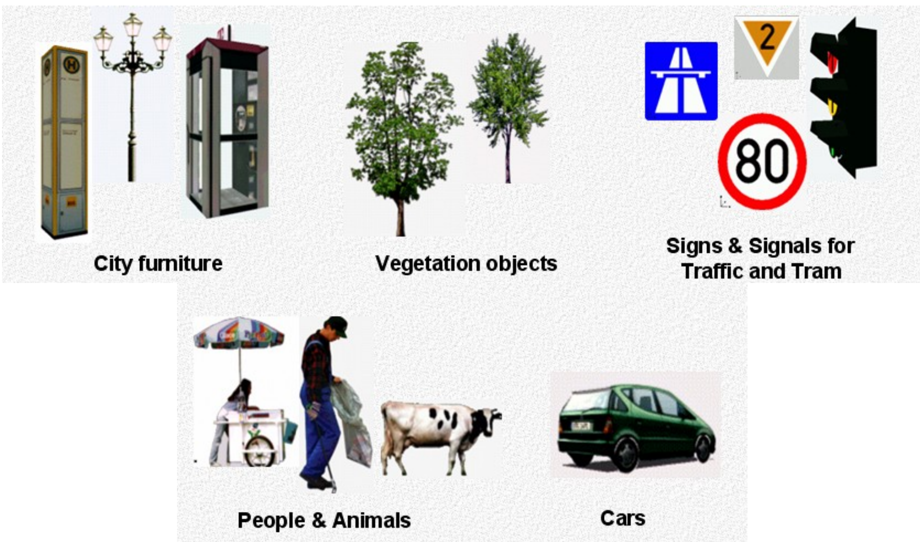


Fig. 6: Examples of prototypic shapes (source: Rheinmetall Defence Electronics).

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 (see 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 10.10).

The current version of CityGML does not include explicit thematic models for bridges, tunnels, and walls. They will be added in a future version. In the meantime, 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 10.11).

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.

Recently, a first ADE for noise pollution simulation has been developed, which is employed in the simulation of environmental noise dispersion according to the Environmental Noise Directive of the European Commission (2002/49/EC). Annex G shows and explains the CityGML Noise ADE as an example.

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 semantically 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.

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 1.0 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 eleven thematic extension modules are introduced by version 1.0 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),
- Building (cf. clause 10.3),
- CityFurniture (cf. clause 10.7),
- CityObjectGroup (cf. clause 10.9),
- Generics (cf. clause 10.10),
- LandUse (cf. clause 10.8),
- Relief (cf. clause 10.2),
- Transportation (cf. clause 10.5),
- Vegetation (cf. clause 10.6),
- WaterBody (cf. clause 10.4), and
- TexturedSurface [deprecated] (cf. clause 9.7).

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 10.11) or by employing the concepts of generic city objects and attributes (cf. chapter 10.10).

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 previous CityGML versions, 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 10.11).

As for version 1.0 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. 7 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.

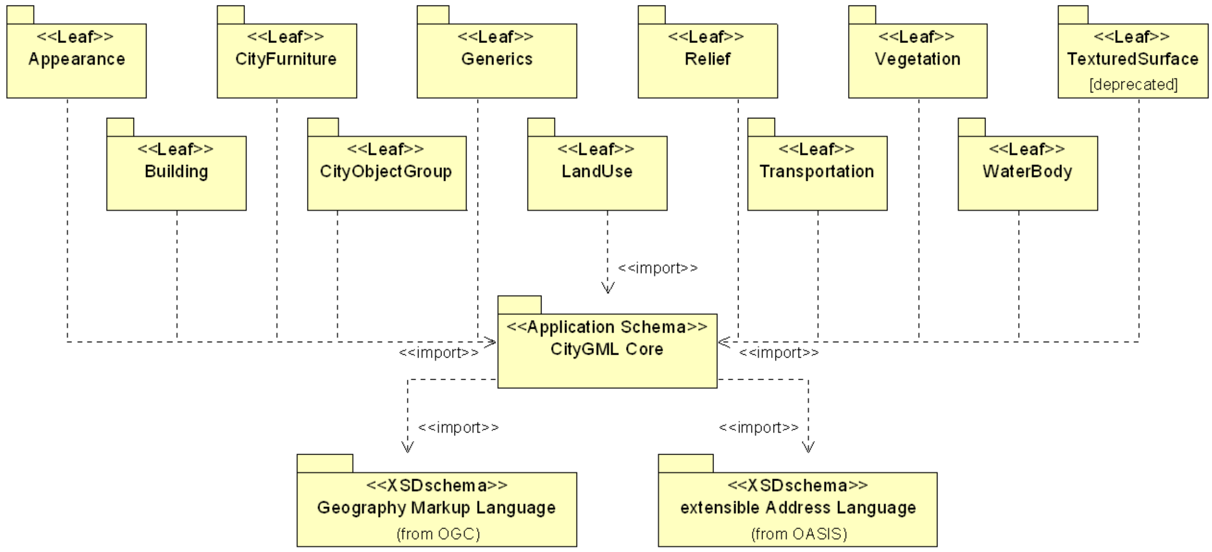


Fig. 7: UML package diagram illustrating the separate modules of CityGML and their schema dependencies. Each extension module (indicated by the leaf packages) further imports the GML 3.1.1 schema definition in order to represent spatial properties of its thematic classes. For readability reasons, the corresponding dependencies have been omitted.

Module name	CityGML Core
XML namespace identifier	http://www.opengis.net/citygml/1.0
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/1.0
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.

Module name	Building
XML namespace identifier	http://www.opengis.net/citygml/building/1.0
XML Schema file	building.xsd
Recommended namespace prefix	bldg
Module description	The *Building* module allows for the representation of thematic and spatial aspects of buildings, building parts, building installations, and interior building structures in four levels of detail (LOD 1 – 4).

Module name	CityFurniture
XML namespace identifier	http://www.opengis.net/citygml/cityfurniture/1.0
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/1.0
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/1.0
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/1.0
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/1.0
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/1.0
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	Vegetation
XML namespace identifier	http://www.opengis.net/citygml/vegetation/1.0
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/1.0
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/1.0
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. An example instance document following this first approach can be found in annex F.1.

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 is shipped with the CityGML schema package and is accessible by the name *CityGML.xsd*.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.citygml.org/citygml/profiles/base/1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.citygml.org/citygml/profiles/base/1.0" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/citygml/appearance/1.0"
            schemaLocation="http://www.citygml.org/citygml/appearance/1.0/appearance.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/building/1.0"
            schemaLocation="http://www.citygml.org/citygml/building/1.0/building.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/cityfurniture/1.0"
            schemaLocation="http://www.citygml.org/citygml/cityfurniture/1.0/cityFurniture.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/cityobjectgroup/1.0"
            schemaLocation="http://www.citygml.org/citygml/cityobjectgroup/1.0/cityObjectGroup.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/generics/1.0"
            schemaLocation="http://www.citygml.org/citygml/generics/1.0/generics.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/landuse/1.0"
            schemaLocation="http://www.citygml.org/citygml/landuse/1.0/landUse.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/relief/1.0"
            schemaLocation="http://www.citygml.org/citygml/relief/1.0/relief.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/transportation/1.0"
            schemaLocation="http://www.citygml.org/citygml/transportation/1.0/transportation.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/vegetation/1.0"
            schemaLocation="http://www.citygml.org/citygml/vegetation/1.0/vegetation.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/waterbody/1.0"
            schemaLocation="http://www.citygml.org/citygml/waterbody/1.0/waterBody.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/texturedsurface/1.0"
            schemaLocation= http://www.citygml.org/citygml/texturedsurface/1.0/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 CityGML modules are necessary.

```
<?xml version="1.0" encoding="UTF-8"?>
<core:CityModel xmlns="http://www.citygml.org/citygml/profiles/base/1.0"
  xmlns:core="http://www.opengis.net/citygml/1.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/1.0"
  xmlns:grp="http://www.opengis.net/citygml/cityobjectgroup/1.0"
  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.citygml.org/citygml/profiles/base/1.0 http://www.citygml.org/citygml/profiles/base/1.0/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>1000</bldg:function>
      <bldg:yearOfConstruction>1985</bldg:yearOfConstruction>
      <bldg:roofType>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. 8 and Fig. 9. 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 GML 3 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. 8). 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.

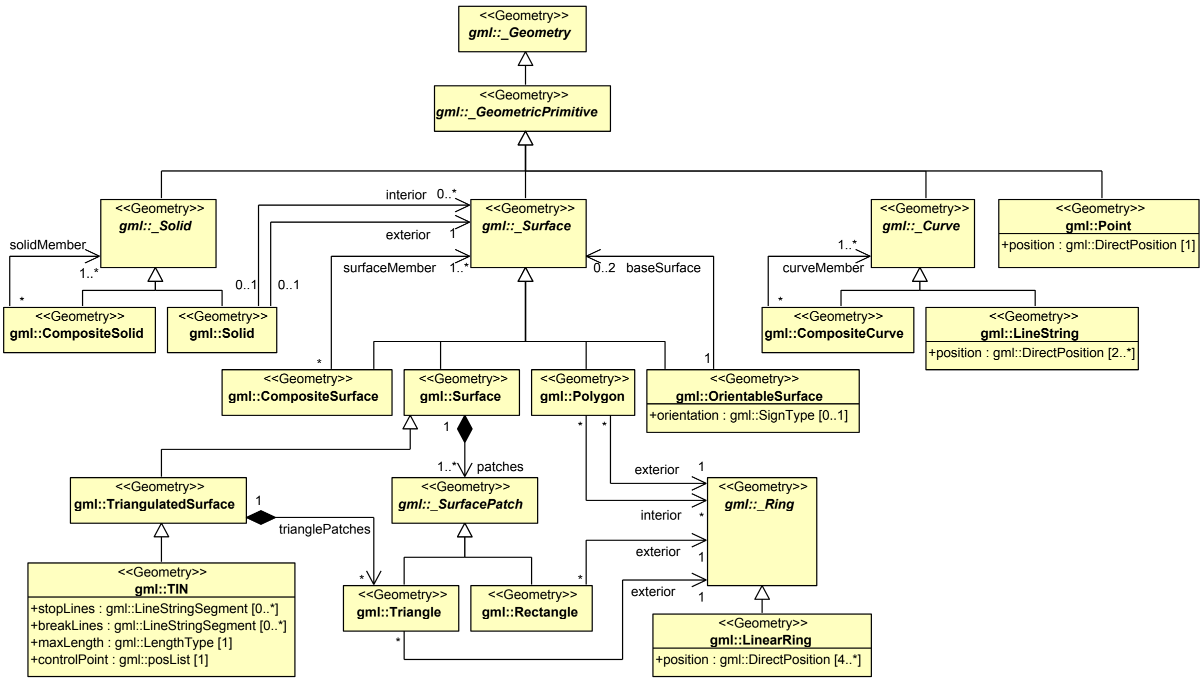


Fig. 8: UML diagram of CityGML's geometry model (subset and profile of GML3): Primitives and Composites.

Combined geometries can be aggregates, complexes or composites of primitives (see illustration in Fig. 10). In 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* or a *MultiSolid* (see Fig. 9). 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. 8).

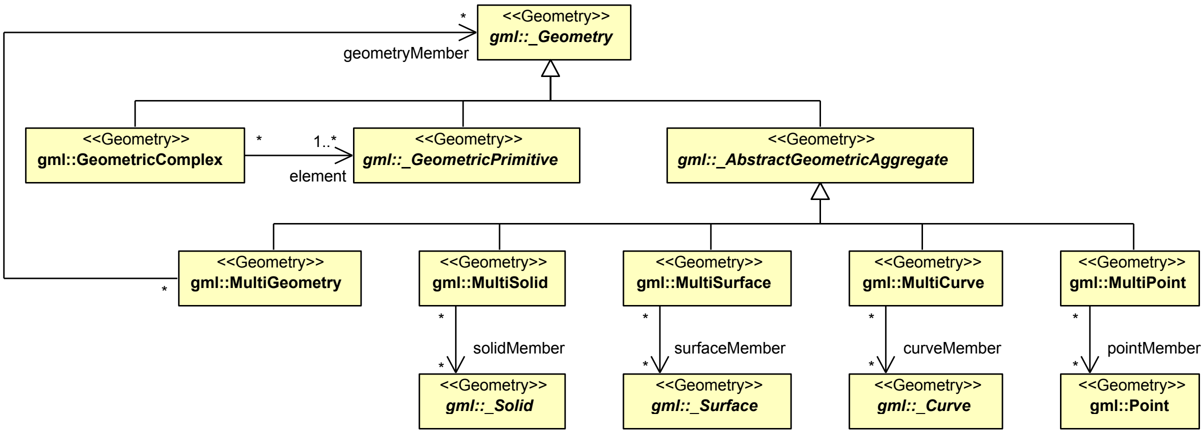


Fig. 9: UML diagram of CityGML's geometry model: Complexes and Aggregates

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.

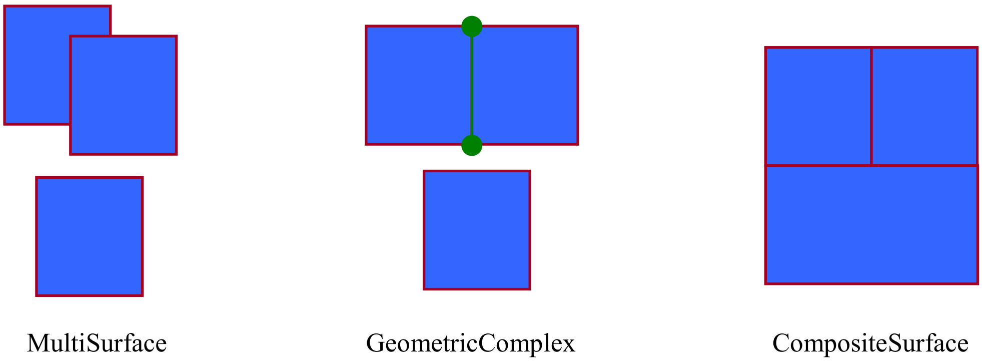


Fig. 10: Combined geometries.

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. 11, 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 F.1.

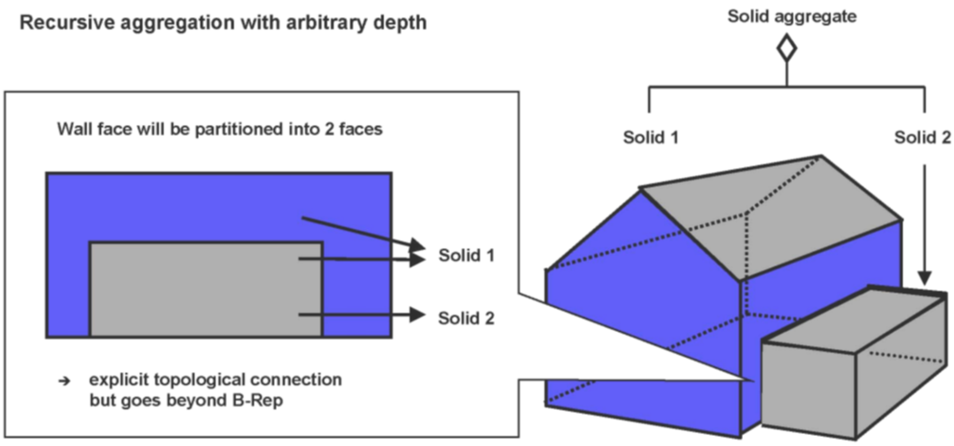


Fig. 11: Recursive aggregation of objects and geometries in CityGML (graphic: IGG Uni Bonn).

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

```
<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 xlink:href="#wallSurface4711"/>
    …
  </bldg:lod2Solid>
  …
</bldg:Building>
```


8.2 *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 vegetation model, for city furniture and generic objects (see chapters 10.6, 10.7 and 10.10). 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. 12. 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 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 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.

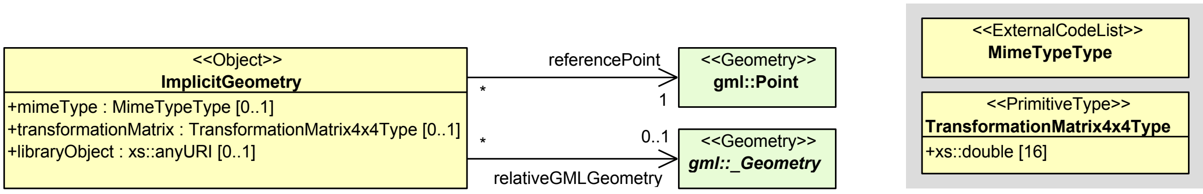


Fig. 12: UML diagram of *ImplicitGeometries*. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the *CityGML Core* module.

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. 40).

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/1.0. 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="MimeTypeType" 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:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="ImplicitGeometry"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

8.2.1 External code lists

The ImplicitGeometry model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.4):

- MimeTypeType

8.2.2 Example CityGML datasets

An example for an implicit geometry is given by the following city furniture object (cf. chapter 10.7), which is represented by a geometry in LOD2:

```
<frn:CityFurniture>
  <frn:class>1000</frn:class>   <!-- "traffic"; declared in external code list (CityFurnitureClassType) in annex C.2 -->
  <frn:function>1080</frn:function>   <!-- "traffic light"; declared in external code list (CityFurnitureFunctionType) in annex C.2 -->
  <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 external code lists proposed in annex C.2) 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>
  <frn:class>1000</frn:class>   <!-- "traffic"; declared in external code list (CityFurnitureClassType) in annex C.2 -->
  <frn:function>1080</frn:function>   <!-- "traffic light"; declared in external code list (CityFurnitureFunctionType) in annex C.2 -->
  <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.

8.2.3 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 using the *mimeType* property (type: *MimeTypeType*).

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. 13, for XML Schema definition see annex A.2.

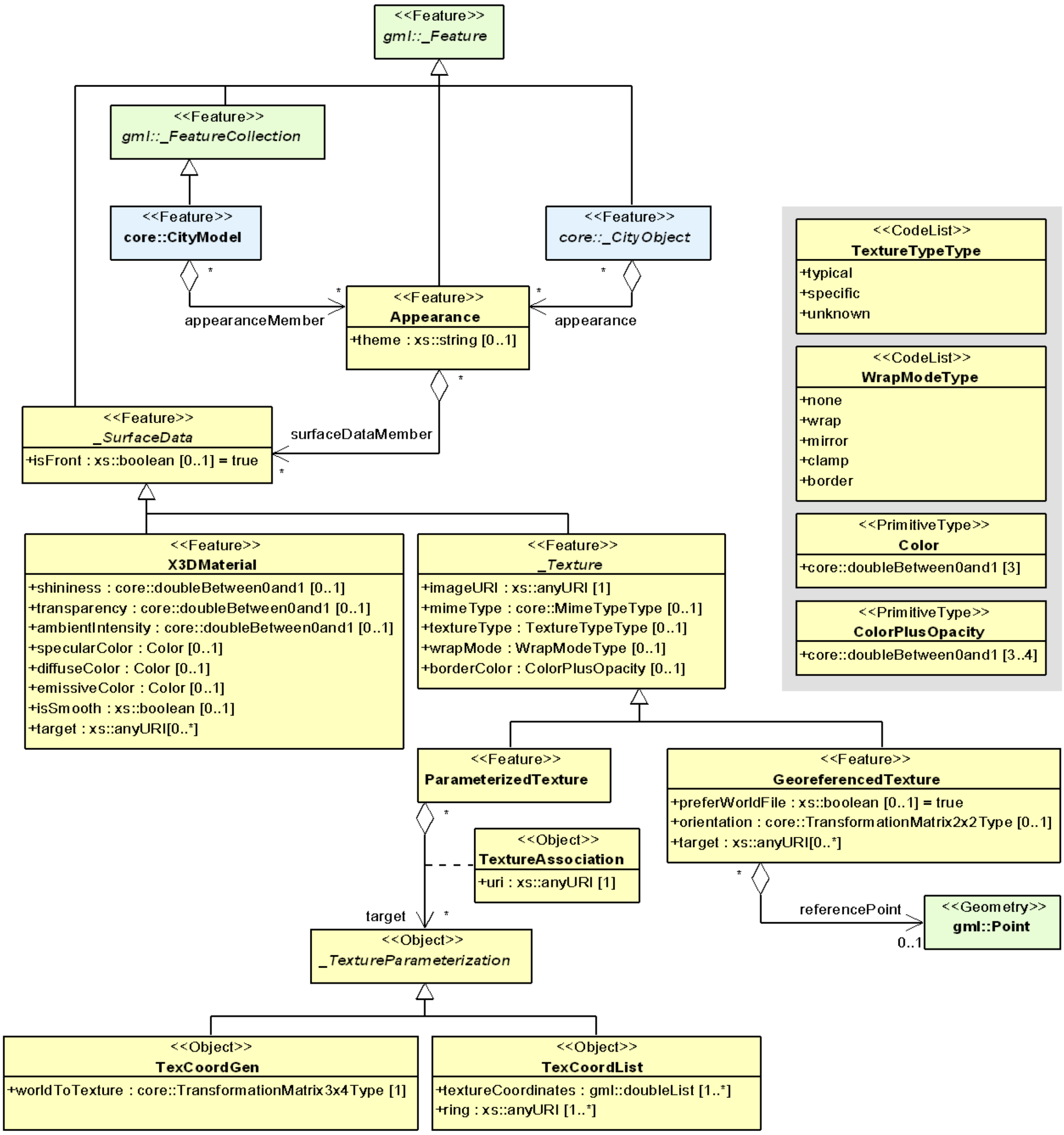


Fig. 13: UML diagram of CityGML's appearance model. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Appearance* module.

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/1.0. 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:Abstract-SurfacePatchType* 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:Multi-Surface* objects. The exclusion of *gml:AbstractSurfacePatchType* as valid target type results from its standard as a root class without *gml:AbstractGMLType* being its parent class. 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.

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 10.11). The hooks for application specific extensions are realized by the elements "*_GenericApplicationPropertyOf…*".

AppearanceType, 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="_GenericApplicationPropertyOfAppearance" type="xs:anyType" abstract="true"/>
<!-- ====================================================================================== -->
<xs:complexType name="AppearancePropertyType">
  <xs:complexContent>
    <xs:extension base="gml:FeaturePropertyType">
      <xs:sequence minOccurs="0">
        <xs:element name="Appearance" type="AppearanceType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

appearanceMember, appearance

```
<xs:element name="appearanceMember" type="AppearancePropertyType" 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 *apperance*. The additional property *appearance* is injected into *_CityObject* using CityGML's *Application Domain Extension* mechanism (cf. chapter 10.11). 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">
  <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. 14 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.

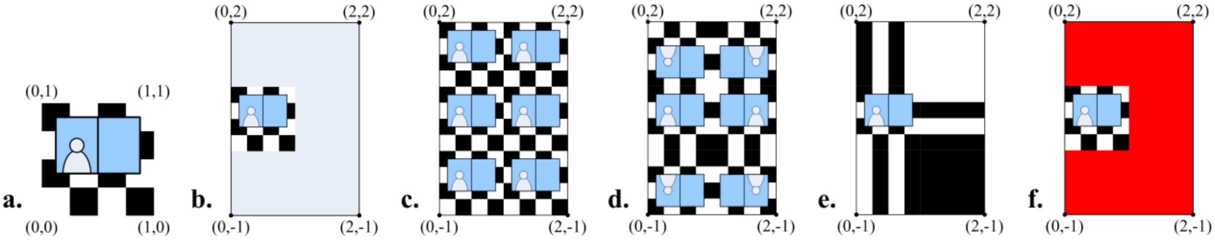


Fig. 14: A texture (a) applied to a facade using different wrap modes: (b) none, (c) wrap, (d) mirror, (e) clamp and (f) border. The border color is red. The numbers denote texture coordinates (image: Hasso-Plattner-Institute).

AbstractTextureType, _Texture, WrapModeType, TextureTypeType

```
<xs:complexType name="AbstractTextureType">
  <xs:complexContent>
    <xs:extension base="AbstractSurfaceDataType">
      <xs:sequence>
        <xs:element name="imageURI" type="xs:anyURI"/>
        <xs:element name="mimeType" type="core:MimeTypeType" 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. The mapping function must be known for each surface geometry object to receive texture.

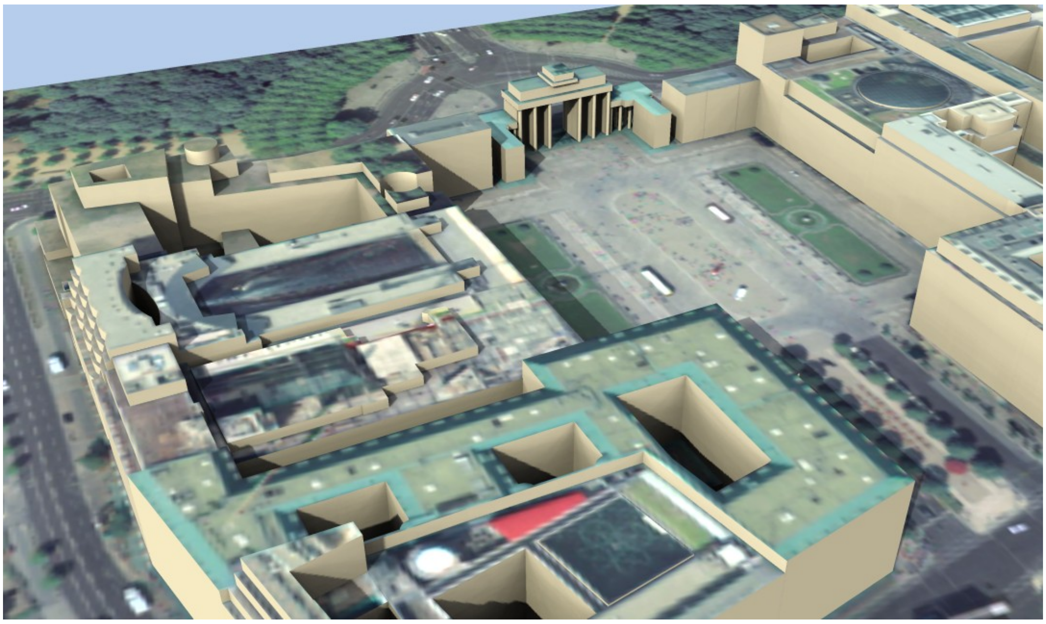


Fig. 15: A georeferenced texture applied to ground and roof surfaces (source: Senate of Berlin, Hasso-Plattner-Institute).

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. 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 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.

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. 15 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 *Tex-CoordGen*).

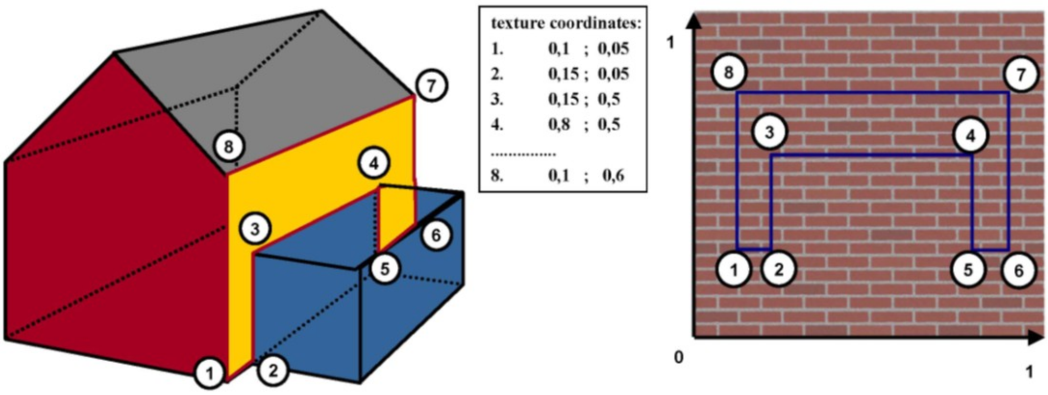


Fig. 16: Positioning of textures using texture coordinates (image: IGG Uni Bonn).

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. 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.

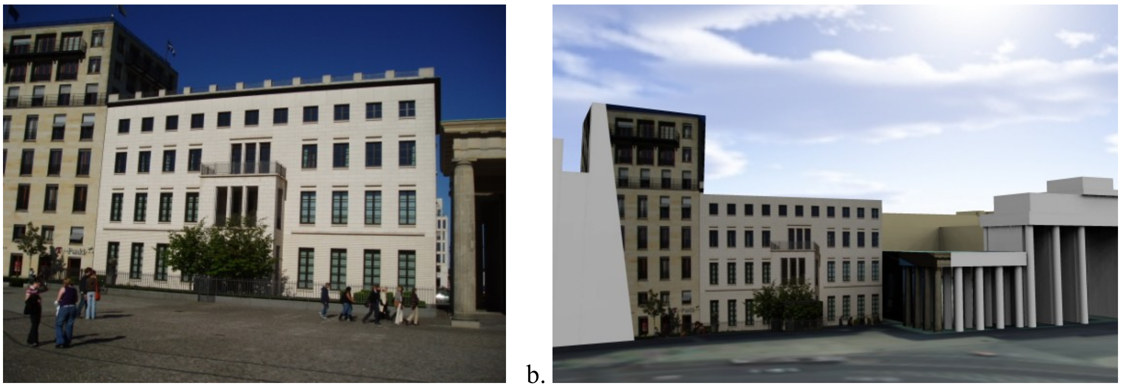


Fig. 17: Projecting a photograph (a) onto multiple facades (b) using the *worldToTexture* transformation. The photograph does not cover the left facade completely. Thus, the texture appears to be clipped. Texture wrapping is set to "*none*" (source: Senate of Berlin, Hasso-Plattner-Institute).

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. 17). 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. 18 sketches this setting.

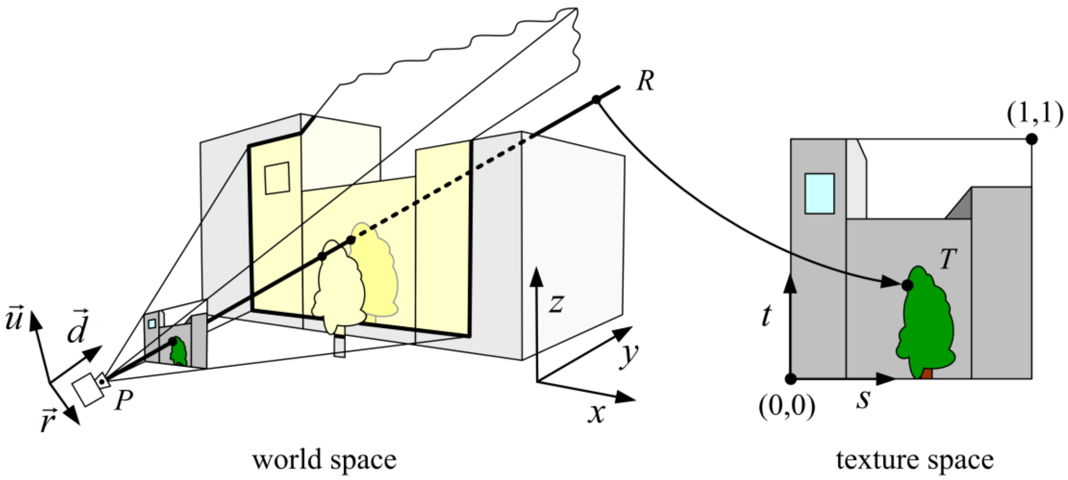


Fig. 18: Projective texture mapping. All points on a ray *R* starting from the projection center *c* are mapped to the same point *T* in texture space (image: Hasso-Plattner-Institute, IGG TU Berlin).

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:

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

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 1.0.0.

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 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. 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 _ *Texture-Parameterization* 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*.

9.7 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 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. 19 depicts the UML diagram, for XML schema definition see annex A.12.

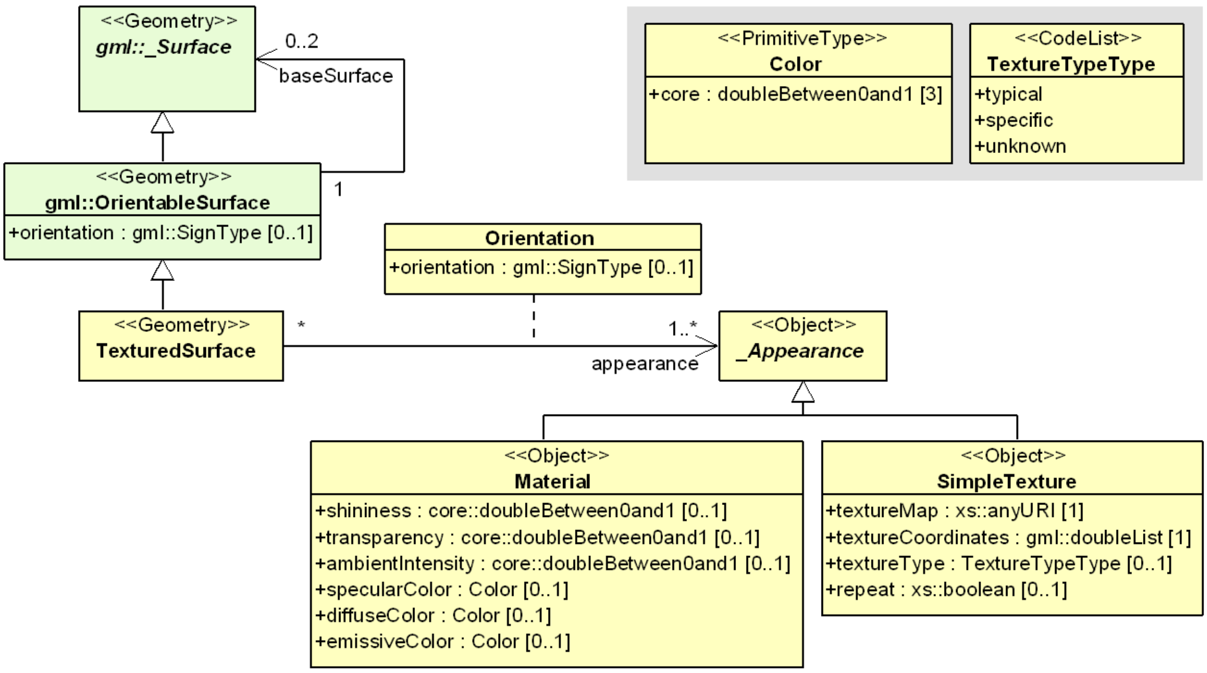


Fig. 19: UML diagram of CityGML's material model. Please note, that this approach for appearance modelling has been marked as deprecated and is expected to be removed in future CityGML versions. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix 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. 16). 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 upper 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/1.0. Within the XML Schema definition of the *Textured-Surface* module, this URI is also used to identify the default namespace.

9.7.1 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 Texture. 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 of the appearance property element, which refers to the corresponding sign 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>
```

9.7.2 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 10.10 and chapter 10.11).

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 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 1.0 of CityGML, the following eleven thematic extension modules are defined: *Appearance*, *Building*, *CityFurniture*, *CityObjectGroup*, *Generics*, *LandUse*, *Relief*, *Transportation*, *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. 20 illustrates CityGML's core module, for XML Schema definition see below and annex A.1.

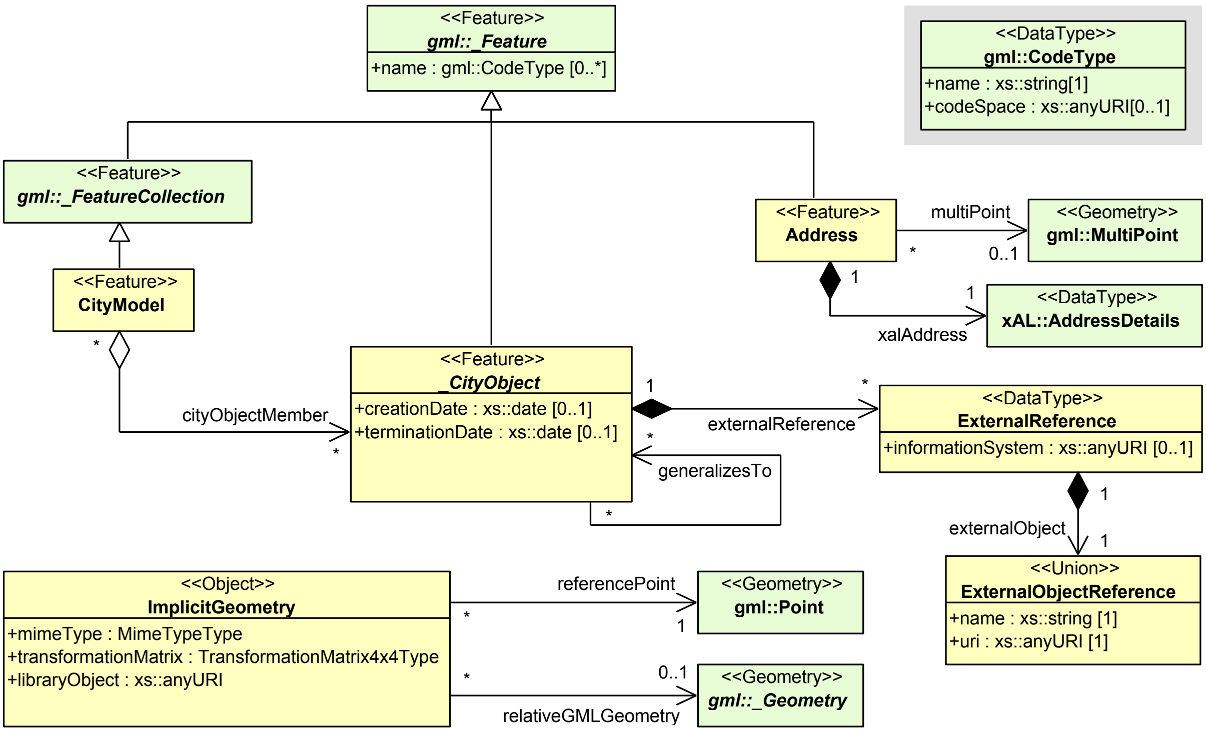


Fig. 20: UML diagram of CityGML's core module. The bracketed numbers following the attribute names denote the attribute's multiplicity: the minimal and maximal number of occurrences of the attribute per object. For example, a *name* is optional (0) in the class *_Feature* or may occur multiple times (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. *_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) 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* denotes the intended purpose or usage of the object, such as hotel or shopping centre for a building, while the attribute *usage* defines its real or actual usage. Possible values for the attributes *class*, *function*, and *usage* are specified in external code lists following the *Simple Dictionary Profile* of GML 3.1.1 (cf. chapter 6.6). CityGML provides predefined external code lists containing feasible attribute values (cf. Annex C). However, these code lists are proposals and 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. 21. The subclasses of *_CityObject* comprise the different thematic fields of a city model covered by separate CityGML extension modules: the terrain, buildings, 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. 21 are preceded with 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).

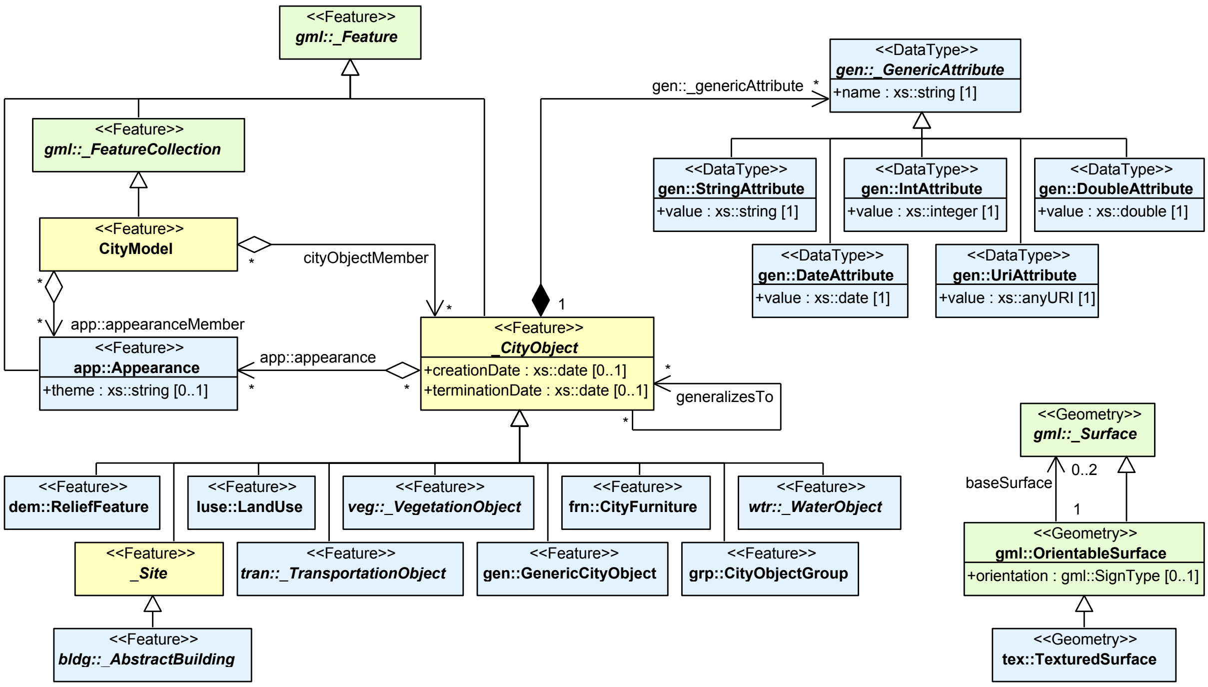


Fig. 21: CityGML's top level class hierarchy. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the *CityGML Core* module.

The classes *GenericCityObject* and *_GenericAttribute* defined within CityGML's *Generics* module (cf. chapters 6.11 and 10.10) 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 will be completed by further subclasses like tunnel, bridge, excavation, wall or embankment. At present, the class *GenericCityObject* may 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 10.11). 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 *CityObjects* 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 10.11). 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. 21. 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/1.0. 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 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, facilities, etc. Future extension of CityGML (e.g. bridges or tunnels) 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

GeneralizationRelationType

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

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 explicitly be 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 address 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:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="Address"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</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 F.1.


```xml
<bldg:Building>
  …
  <bldg:address>
    <Address>
      <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>
      </xalAddress>
    </Address>
  </bldg:address>
</bldg:Building>
```

```xml
<bldg:Building>
  …
  <bldg:address>
    <Address>
      <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>
      </xalAddress>
    </Address>
  </bldg:address>
</bldg:Building>
```

10.1.5 External code lists

The ImplicitGeometry model of CityGML’s core module introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.4):

- MimeTypeType

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 8.2.3) 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. 22 depicts the UML diagram, for XML schema definition see annex A.8). 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 *(MasspointRelief*). The four types are implemented by the corresponding GML3 classes: grids by *RectifiedGridCoverages,* break lines by *Curves,* mass points by *Points* and TINs either by *TriangulatedSurfaces* or by GML3 *TINs*. In case of *TriangulatedSurfaces*, the triangles are given explicitly while in case of GML3 *TINs* 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.

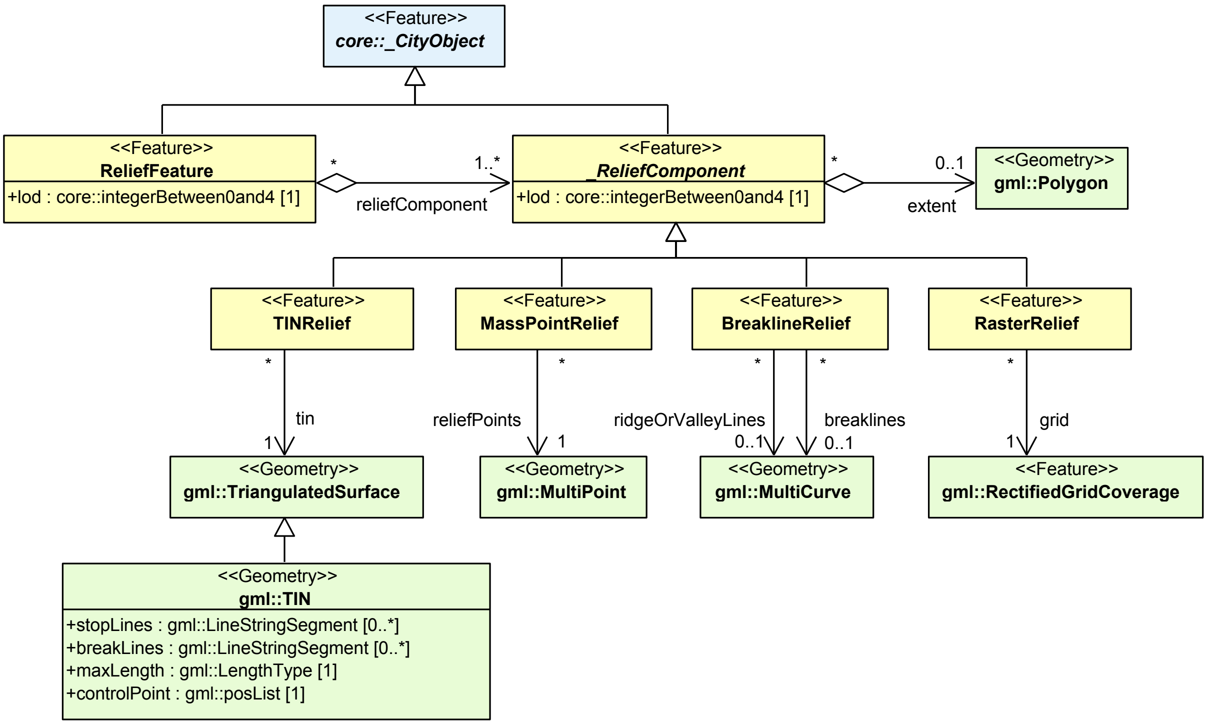


Fig. 22: UML diagram of the Digital Terrain Model in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Relief* module.

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. 23). 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.

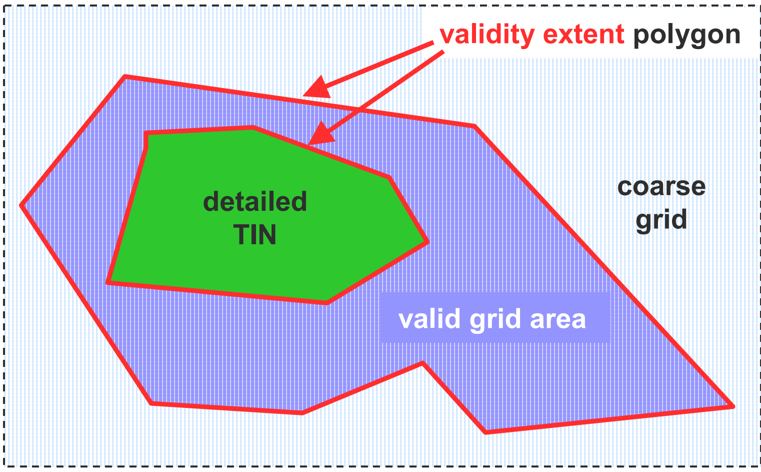


Fig. 23: Nested DTMs in CityGML using validity extent polygons (graphic: IGG Uni Bonn).

Accuracy and resolution of the DTM are not necessarily dependent on those of the building model. 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* (see chapter 6.5). The TIC can be used like break lines to adjust the DTM to different building models 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 building model.

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/1.0. 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:complexContent>
```



```
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="_ReliefComponent"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</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:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="gml:TriangulatedSurface"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</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:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="gml:RectifiedGridCoverage"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</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 the most detailed thematic concept of CityGML. It allows for the representation of thematic and spatial aspects of buildings, building parts and installations in four levels of detail, LOD1 to LOD4. The building model of CityGML is defined by the thematic extension module *Building* (cf. chapter 7). Fig. 24 provides examples of 3D city models for each LOD.

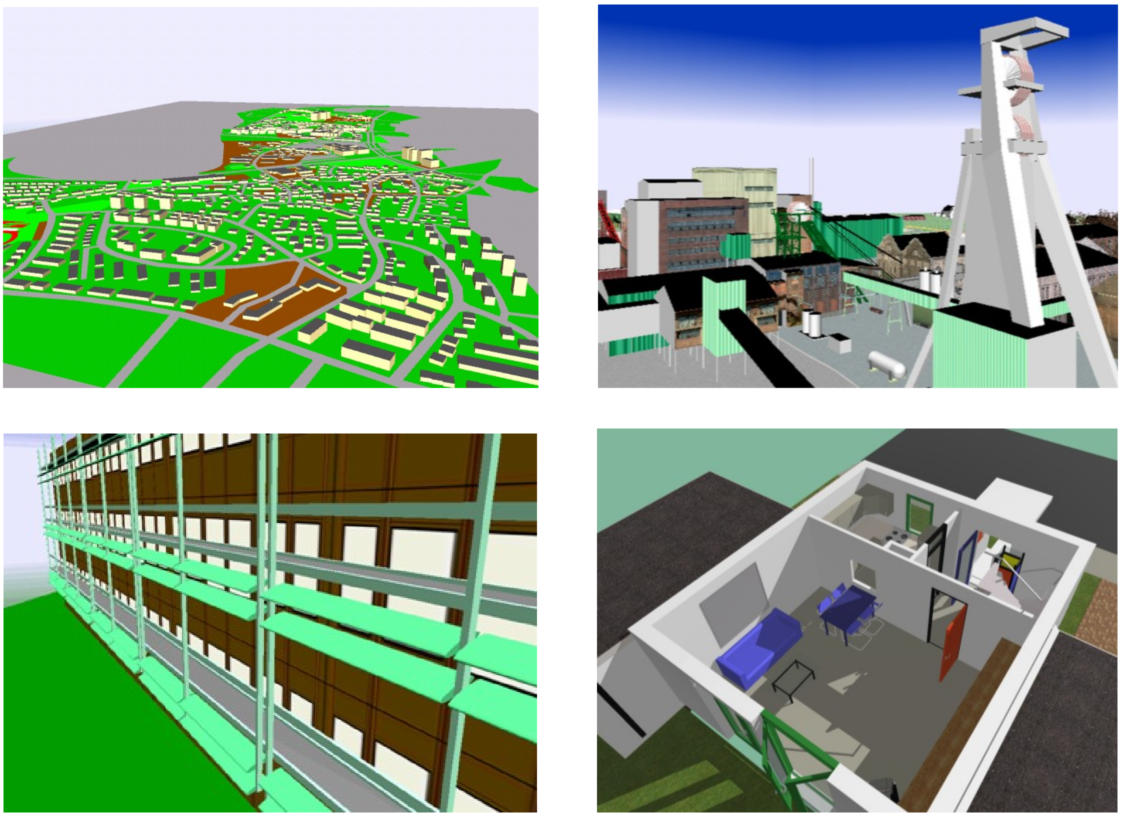


Fig. 24: Examples for city or building models in LOD1 (upper left), LOD2 (upper right), LOD3 (lower left), and LOD4 (lower right) (source: District of Recklinghausen, m-g-h ingenieure+architekten GmbH).

The UML diagram of the building model is depicted in Fig. 25, for the XML schema definition see annex A.3 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 *GenericAttributes* provided by the CityGML *Generics* module (cf. chapter 10.10).

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* (see chapter 6.8). The main building of the complex can be denoted by providing “main building” as the role name of the corresponding object.

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, *Addresses* can be assigned to *Buildings* or *BuildingParts*.

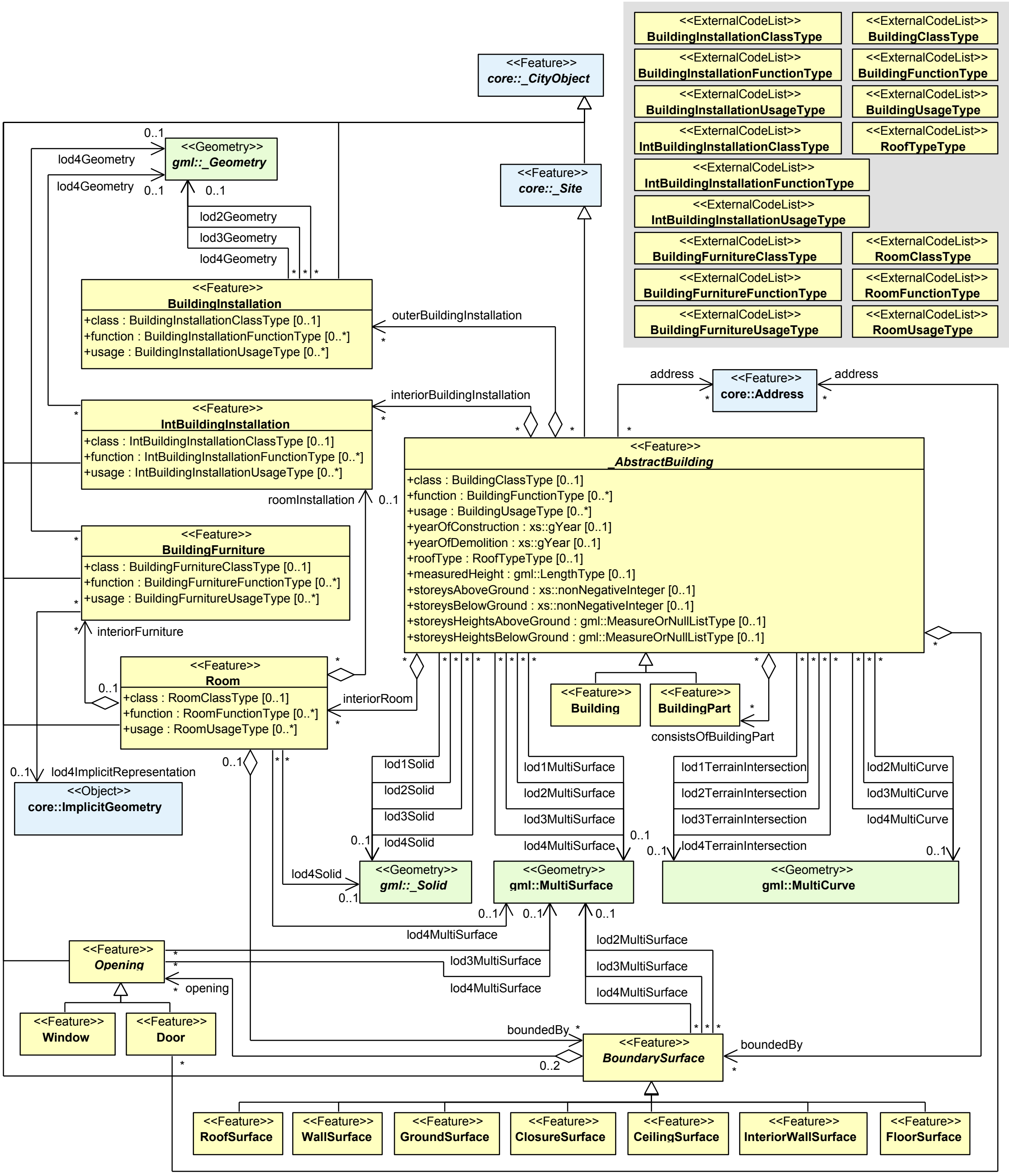


Fig. 25: UML diagram of CityGML's building model. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Building* module.

The geometric representation and semantic structure of an *_AbstractBuilding* is shown in Fig. 25. The model is successively refined from LOD1 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 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 LOD1, a building model consists of a geometric representation of the building volume. Optionally, a *MultiCurve* representing the *TerrainIntersectionCurve* (see chapter 6.5) can be specified. This geometric representation is refined in LOD2 by additional *MultiSurface* and *MultiCurve* geometries, used for modelling architectural details like a roof overhang, 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*) 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. 25).

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 (see 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* referenced to external code lists (chapter 10.3.7 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*. An example CityGML dataset containing simple buildings can be found in annex F.1.

XML namespace

The XML namespace of the CityGML *Building* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/building/1.0. 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 *BuildingParts* (see Fig. 26). 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. 26). A *BuildingPart* object should be uniquely related to exactly one building or building part object.

Building with two building parts (represented as one *Building* feature and one included *BuildingPart* feature)

Building consisting of one part (represented as one *Building* feature)

Fig. 26: 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="BuildingClassType" minOccurs="0"/>
        <xs:element name="function" type="BuildingFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="BuildingUsageType" 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="RoofTypeType" 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="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"/>
```



```xml
			<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 on different levels of detail. The attributes describe:

a) The classification of the building or building part (*class*), the different functions (*function*), and the usage (*usage*). The permitted values for these property types are specified in a separate XML file, using the dictionary concept of GML3.
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 the *RoofTypeType* are specified in a separate XML-File, using the dictionary concept of GML.
d) The measured relative height (*measuredHeight*) of the building or building part ridge line (highest point).
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. 27, showing the same building in four different LODs. The class *_AbstractBuilding* has a number of properties which are associated with certain LODs.

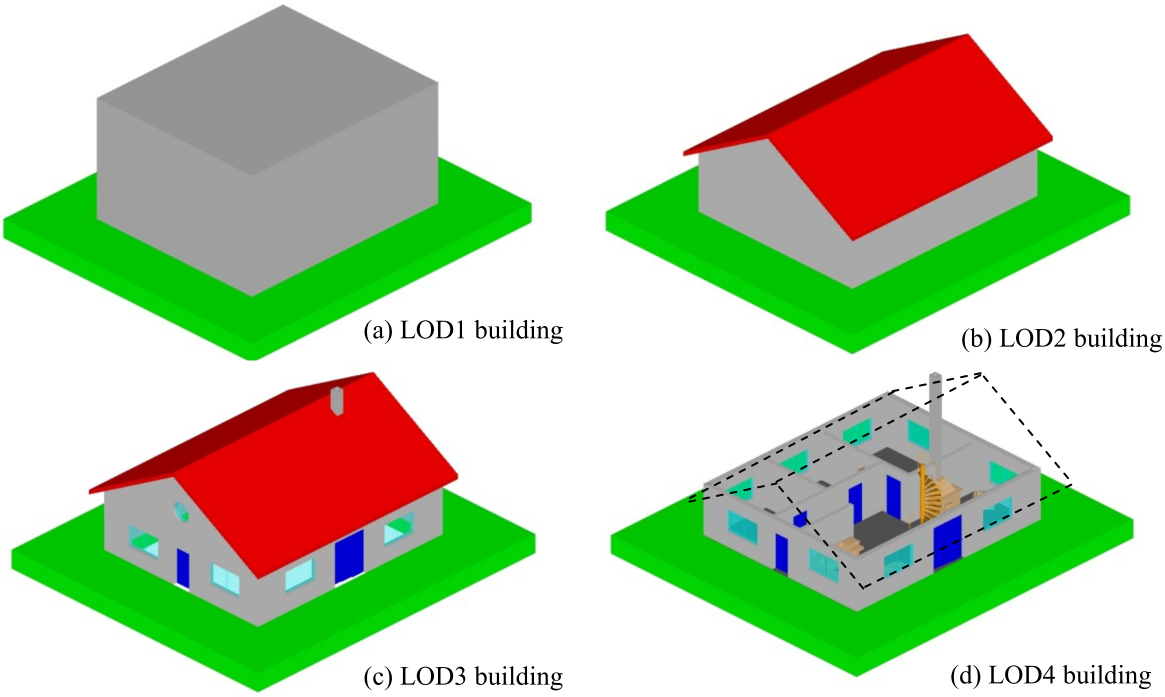


Fig. 27: Building model in LOD1 – LOD4 (source: Research Center Karlsruhe).

Tab. 5 shows the correspondence of the different geometric and semantic themes of the building model to LODs. In each LOD, the volume of a building can be expressed by a *SolidGeometry* and/or a *MultiSurfaceGeometry*. 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 all four LODs. The TIC can – but does not have to – build closed rings around the building or building parts.

In LOD1 (see Fig. 27 a), the different structural entities of a building are aggregated to simple blocks 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 *SolidGeometry* and/or *MultiSurfaceGeometry*, it can also be composed of semantic objects. The base class for all objects semantically structuring the building shell is *_BoundarySurface* (see chapter 10.3.2), which is associated with a *MultiSurfaceGeometry*. 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 *MultiSurfaceGeometry* of the *_BoundarySurfaces*.

Geometric / semantic theme	**Property type**	**LOD 1**	**LOD 2**	**LOD 3**	**LOD 4**
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*		•	•	•
BoundarySurfaces (chapter 10.3.2)	*_BoundarySurfaceType*		•	•	•
Outer building installations (chapter 10.3.3)	*BuildingInstallationType*		•	•	•
Openings (chapter 10.3.4)	*_OpeningType*			•	•
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* (see chapter 10.3.3). Typical candidates for this class are chimneys (see Fig. 27 c), dormers (see Fig. 26), balconies, outer stairs, or antennas. *BuildingInstallations* may only be included in LOD2 models, if their extents exceed the 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. 8 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* (see chapter 10.3.2). In LOD3, openings in a building corresponding with windows or doors (see Fig. 27 c) are modelled by the (abstract) class *_Opening* and the derived classes *Window* and *Door* (see chapter 10.3.4).

With respect to the exterior building shell, the LOD4 data model is identical to that of LOD3. But LOD4 provides for a possibility to describe the interior structure of a building with the classes *IntBuildingInstallation* and *Room* (see 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 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 and the visible surface of a room. 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, ClosureSurface, FloorSurface, InteriorWallSurface,* and *CeilingSurface* are derived. The thematic classification of building surfaces is illustrated in Fig. 28 and subsequently specified.

For each LOD between 2 and 4, the geometry of a *_BoundarySurface* may be defined by a different *MultiSurfaceGeometry*. In LOD2, this surface geometry must be simply connected, which means that the components of the *MultiSurface* (e.g. *gml:Polygon*) must not have inner holes (*gml:interior*).

In LOD3 and LOD4, a *_BoundarySurface* may reference *_Openings* (see 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 *MultiSurfaceGeometry,* these *_Openings* must be represented as holes within that surface. A hole is represented by an interior ring within the corresponding surface geometry object. 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. However, 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). 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. 28 on the right).

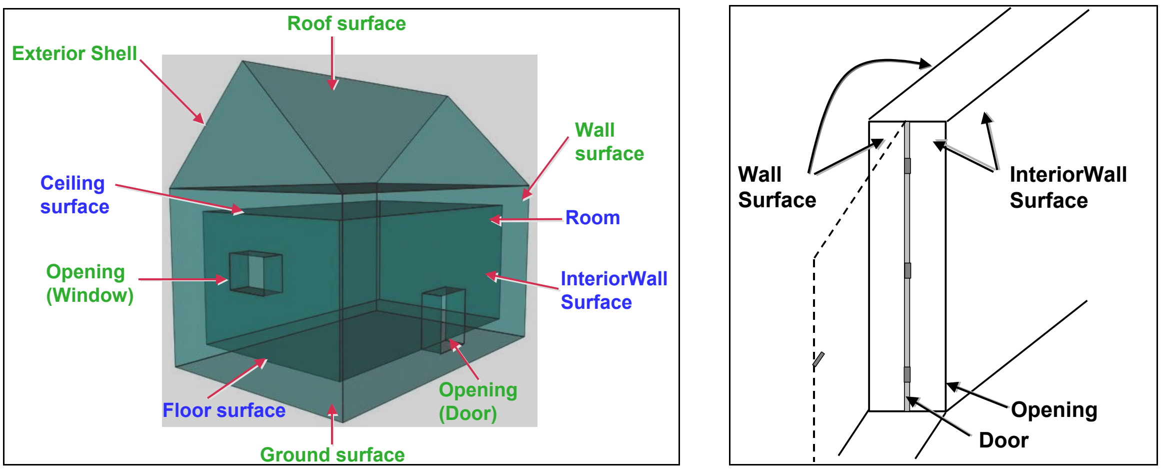


Fig. 28: Classification of *BoundarySurfaces* (left), in particular for *Openings* (right) (graphic: IGG Uni Bonn).

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 *BuildingInstallations*.

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 visible from the outside are modelled by the class *WallSurface*.

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.

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* (see 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.

10.3.3 Outer building installations

BuildingInstallationType, BuildingInstallation

```
<xs:complexType name="BuildingInstallationType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="BuildingInstallationClassType" minOccurs="0"/>
        <xs:element name="function" type="BuildingInstallationFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="BuildingInstallationUsageType" 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 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 is specified in a GML dictionary. For the geometrical representation of a *BuildingInstallation*, an arbitrary geometry object from the GML subset shown in Fig. 8 can be used. A *BuildingInstallation* object should be uniquely related to exactly one building or building part object.

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 walls. Openings only exist in models of LOD3 or LOD4. Each *_Opening* is associated with a *MultiSurfaceGeometry*.

WindowType, Window

```
<xs:complexType name="WindowType">
  <xs:complexContent>
    <xs:extension base="AbstractOpeningType">
```



```xml
        <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

```xml
<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 *AddressPropertyType* is defined within the CityGML core module (cf. chapter 10.1.4) .

10.3.5 Building interior

RoomType, Room

```xml
<xs:complexType name="RoomType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="RoomClassType" minOccurs="0"/>
        <xs:element name="function" type="RoomFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="RoomUsageType" 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*, see chapter 10.3.2).

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. 29).

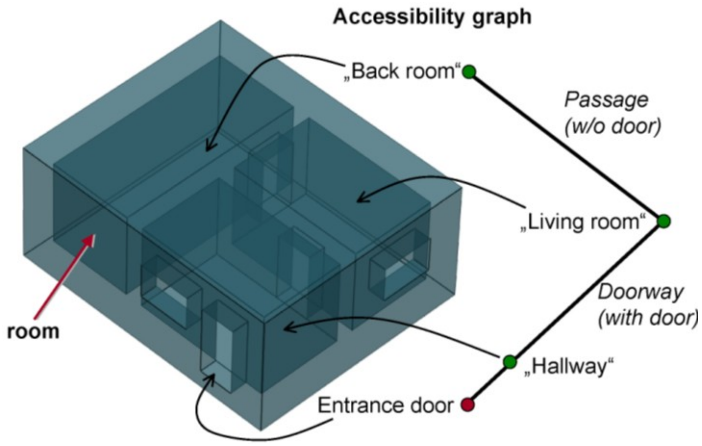


Fig. 29: Accessibility graph derived from topological adjacencies of room surfaces (graphic: IGG Uni Bonn).

BuildingFurnitureType, BuildingFurniture

```
<xs:complexType name="BuildingFurnitureType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="BuildingFurnitureClassType" minOccurs="0"/>
        <xs:element name="function" type="BuildingFurnitureFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="BuildingFurnitureUsageType" 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.

IntBuildingInstallationType, IntBuildingInstallation

```
<xs:complexType name="IntBuildingInstallationType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
```



```
        <xs:element name="class" type="IntBuildingInstallationClassType" minOccurs="0"/>
        <xs:element name="function" type="IntBuildingInstallationFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="IntBuildingInstallationUsageType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <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*, see 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 is specified in a GML dictionary. For the geometrical representation of an *IntBuildingInstallation*, an arbitrary geometry object from the GML subset shown in Fig. 8 can be used.

10.3.6 Modelling building storeys using CityObjectGroups

CityGML does 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 10.9). 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 *CityGMLGroup* objects has to be used. On the first level, 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.

On the second level, the *CityGMLGroup* objects representing different storeys are grouped itself. By using the generic aggregation concept of *CityGMLGroup,* 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 External code lists

The building model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.1):

- BuildingClassType
- BuildingFunctionType
- BuildingUsageType
- RoofTypeType
- BuildingInstallationClassType
- BuildingInstallationFunctionType

- BuildingInstallationUsageType
- IntBuildingInstallationClassType
- IntBuildingInstallationFunctionType
- IntBuildingInstallationUsageType
- BuildingFurnitureClassType
- BuildingFurnitureFunctionType
- BuildingFurnitureUsageType
- RoomClassType
- RoomFunctionType
- RoomUsageType

10.3.8 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 *lodXSolid* and *lodXMultiSurface*, X ∈ [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.
3. 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,* 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 ∈ [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 building shell may be represented using the *lodXMultiCurve*, X ∈ [2..4], property of *_AbstractBuilding*. This property shall not be used if the building is only represented in LOD1.
5. 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.
6. 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 and, thus, must not have inner holes.

   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*.

7. 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.
8. 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.

Referential integrity

9. 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,* and *ClosureSurface* elements are allowed to be encapsulated or referenced by the *boundedBy* property of *_AbstractBuilding*.
10. 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.
11. 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.
12. 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.
13. 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.

14. 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.
15. 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.
16. 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.
17. 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*.
18. 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.
19. 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.

10.4 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. 30). A dynamic element may be the *WaterSurface* to represent temporarily changing situations of tidal flats.

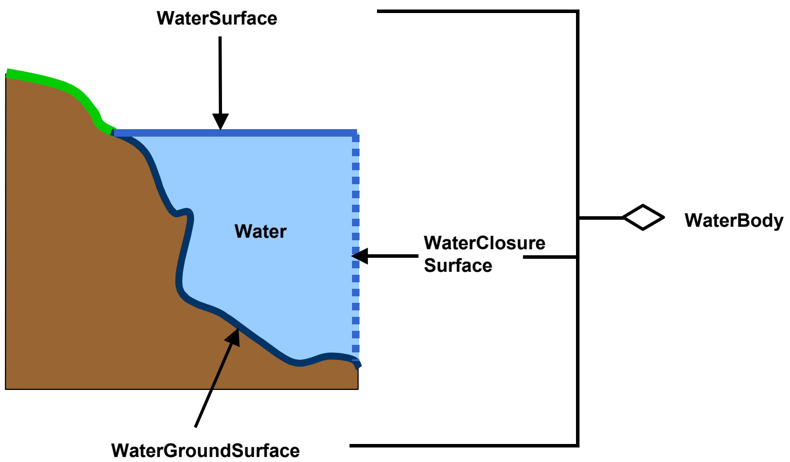


Fig. 30: Illustration of a water body defined in CityGML (graphic: IGG Uni Bonn).

The UML diagram of the water body model is depicted in Fig. 31, for XML schema definition see below and annex A.11. Each *WaterBody* object may have the attributes *class*, *function* and *usage* referencing to external code lists (cf. chapter 10.4.3 and annex C.7). 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 thus of the root class *_CityObject*. The class *_WaterObject* can 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 *External-References* (cf. chapter 6.7) and may be augmented by *GenericAttributes* using CityGML's *Generics* module (cf. chapter 10.10).

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 are defined in the respective external code list.

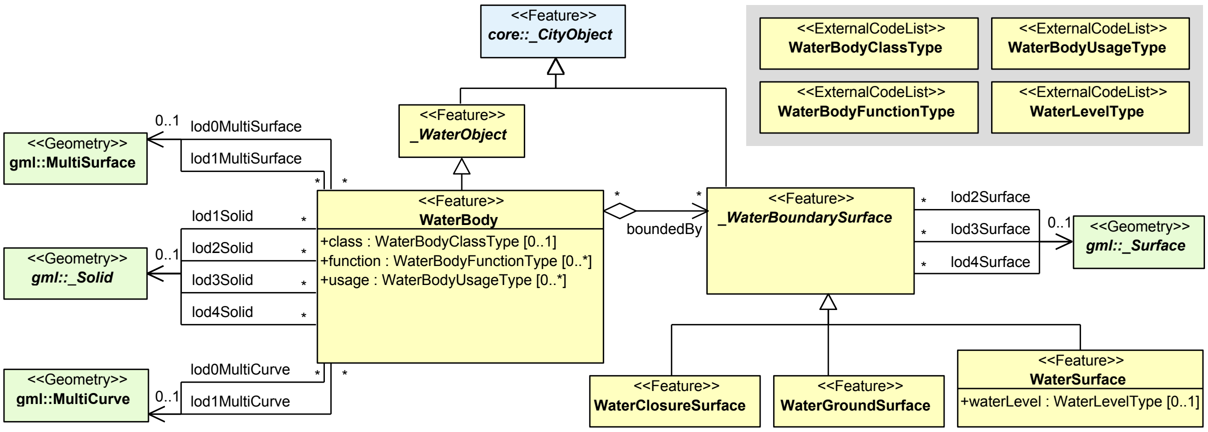


Fig. 31: UML diagram of the water body model in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *WaterBody* module.

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. 31). 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 *Solid* in LOD2 or higher, the surface geometries of the corresponding thematic *Water-ClosureSurface*, *WaterGroundSurface*, and *WaterSurface* objects must coincide with the exterior shell of the *Solid*. This can be ensured, if for one LOD X the respective *lodXSurface* elements (where *X* is between 2 and 4) of *WaterClosureSurface*, *WaterGroundSurface*, and *WaterSurface* do not redundantly define *gml:Polygons,* but instead reference the corresponding polygons (using XLink) within the *CompositeSurface* that defines the exterior shell of the *Solid*.

LOD2 to LOD4 demand a higher grade of detail and therefore any *WaterBody* will 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/1.0. Within the XML Schema definition of the *WaterBody* module, this URI is also used to identify the default namespace.

10.4.1 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" 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="WaterBodyClassType" minOccurs="0"/>
        <xs:element name="function" type="WaterBodyFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="WaterBodyUsageType" 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"/>
```

10.4.2 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. 30).

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* (*WaterLevelType)*, as for example highest flooding event, mean sea level, or minimum water level, given in an external code list. 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 Building”. 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 buildings like sluices, sills, flood barrage or tidal power stations. The use of *WaterGroundSurface* as boundary layer to buildings is relevant in urban situations, where buildings enclose the defined water completely such as fountains and swimming pools. Together, the *WaterSurface* and *WaterGroundSurface* 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="WaterLevelType" 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"/>
```

10.4.3 External code lists

The water bodies model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.7):

- WaterLevelType
- WaterBodyClassType
- WaterBodyFunctionType
- WaterBodyUsageType

10.4.4 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 an additional geometric representation as volume model using the *lodXSolid*, X ∈ [2..4], properties of *WaterBody*, the *_WaterBoundarySurface* elements shall not explicitly define the geometry, but have to reference the according surface components of the *gml:Solid* element of *WaterBody* 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.

10.5 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. 32 illustrates the four different thematic classes.

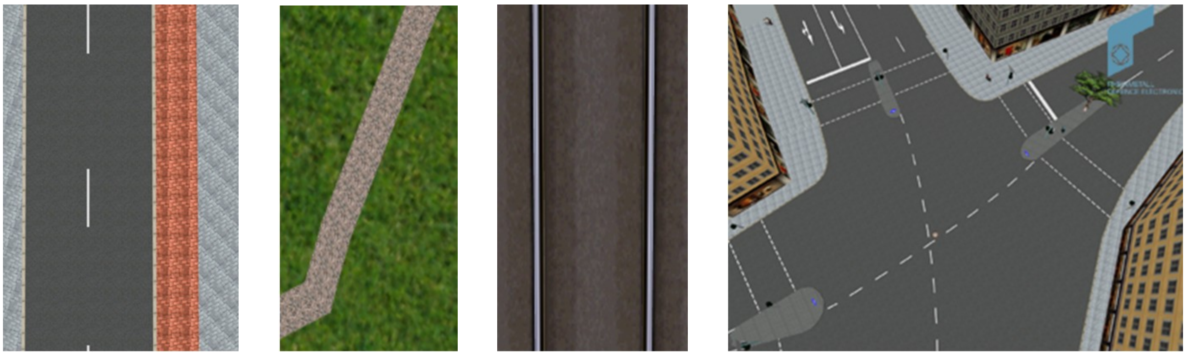


Fig. 32: Representations of *TransportationComplex* (from left to right: examples of road, track, rail, and square) (source: Rheinmetall Defence Electronics).

A *TransportationComplex* is composed of the parts *TrafficArea* and *AuxiliaryTrafficArea*. Fig. 33 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.

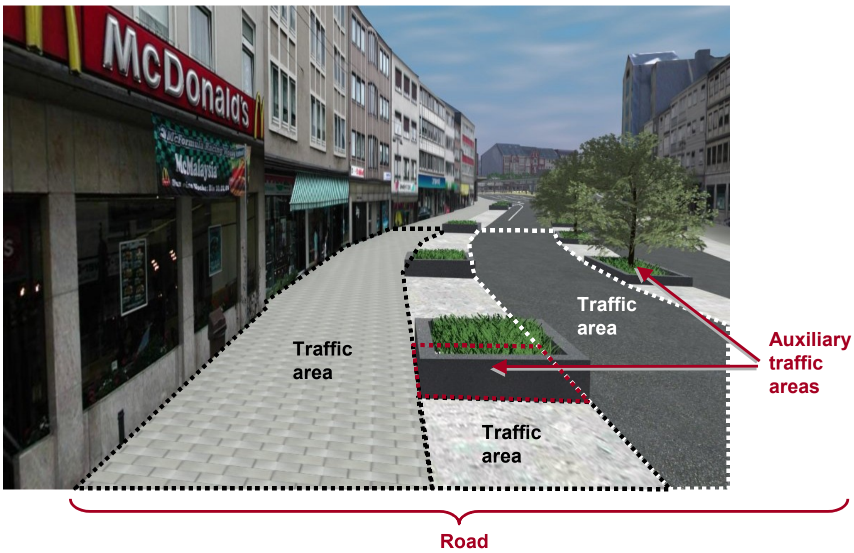


Fig. 33: Example for the representation of a *TransportationComplex* in LOD2 in CityGML: a road, which is the aggregation of *TrafficAreas* and *AuxiliaryTrafficAreas* (source: City of Solingen, IGG Uni Bonn).

Fig. 34 depicts the UML diagram of the transportation model, for XML schema definition see annex A.9.

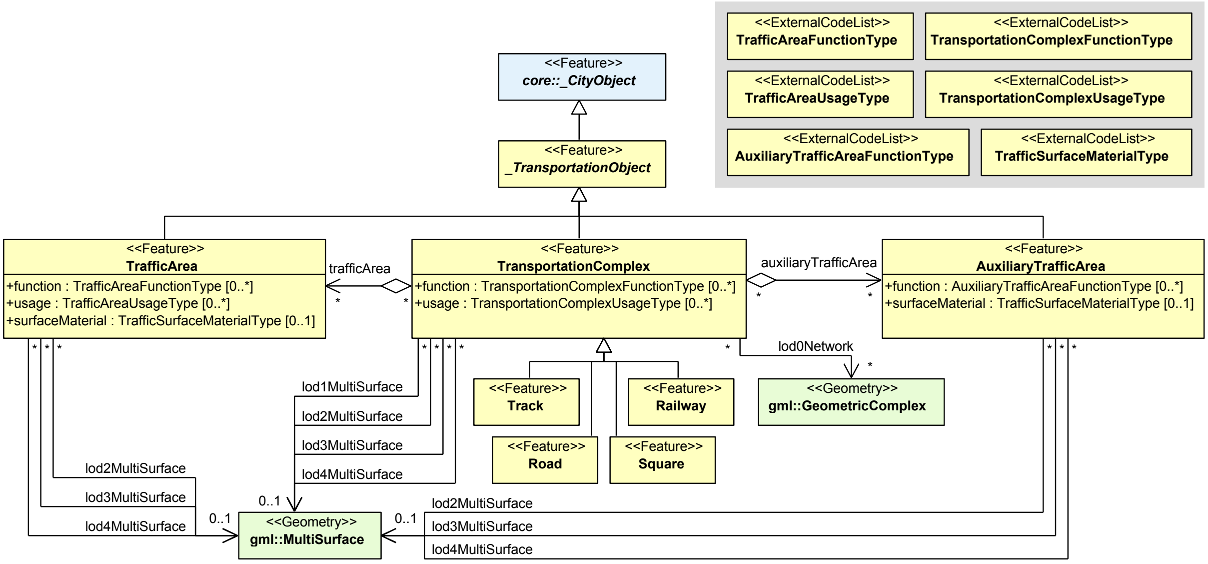


Fig. 34: UML diagram of the transportation model in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Transportation* module.

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 *function* and *usage*, referencing to external code lists (chapter 10.5.4 and annex C.6). The attribute *function* describes the purpose of the object like, for example national motorway, country road, or airport, 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.

In addition every *TrafficArea* may have the attributes *function*, *usage*, and *surfaceMaterial*. The *function* describes, if the object 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 etc.). The *function* attribute of the *AuxiliaryTrafficArea* defines, for example kerbstones, middle lanes, or green areas. The possible values are also specified in external code lists.

The shape of each traffic area is defined by an area geometry. Additional metadata may be defined by using attributes from pre-defined catalogues. This affects the function of the area, the usage and surface material definition for each area. 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 description 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. 35.

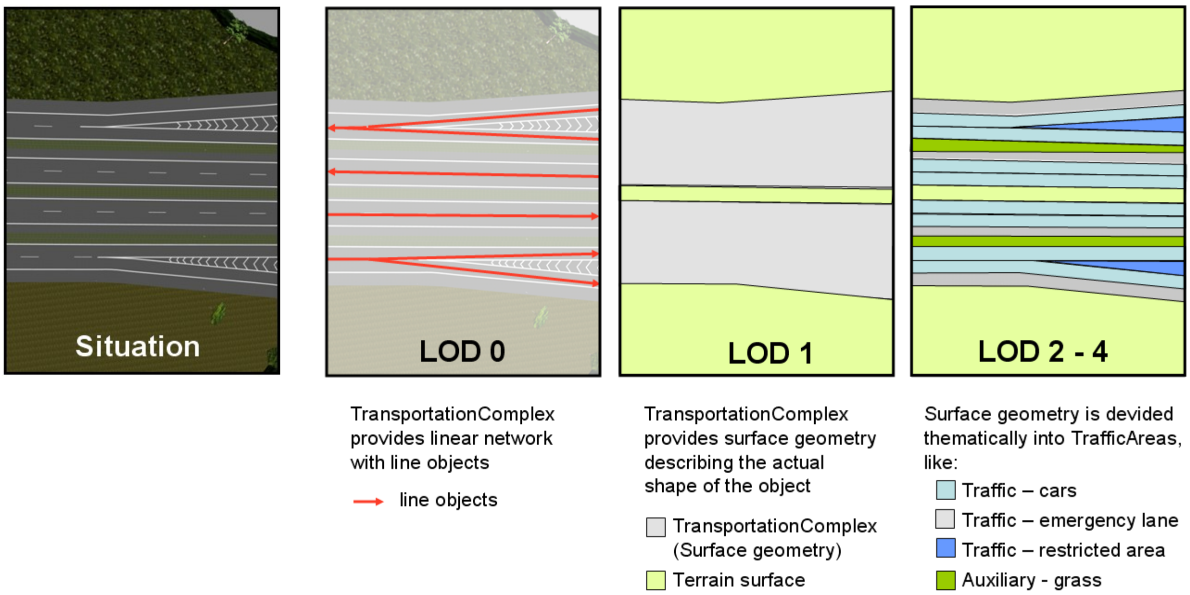


Fig. 35: *TransportationComplex* in LOD0, 1, and 2-4 (example shows part of a motorway) (source: Rheinmetall Defence Electronics).

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, *GenericAttributes* provided by CityGML's *Generics* module (cf. chapter 10.10) 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 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).

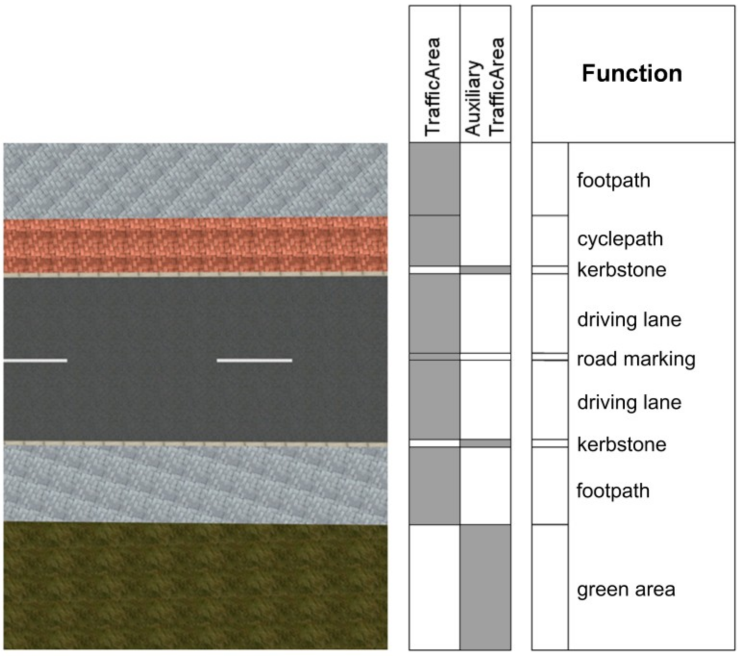


Fig. 36: *TransportationComplex* in LOD 2-4: representation of a road with a complex cross-section profile (example shows urban road) (source: Rheinmetall Defence Electronics).

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. 37: 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/1.0. Within the XML Schema definition of the *Transportation* module, this URI is also used to identify the default namespace.

10.5.1 Transportation complex

AbstractTransportationObjectType, _TransportationObject

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



```xml
    <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" 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

```xml
<xs:complexType name="TransportationComplexType">
  <xs:complexContent>
    <xs:extension base="AbstractTransportationObjectType">
      <xs:sequence>
        <xs:element name="function" type="TransportationComplexFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="TransportationComplexUsageType" 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 describes 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.

10.5.2 Subclasses of transportation complexes

TrackType, Track

```xml
<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 *Transportation-Complex* 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.

10.5.3 Subdivisions of transportation complexes

TrafficAreaType, TrafficArea

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



```
    <xs:extension base="AbstractTransportationObjectType">
      <xs:sequence>
        <xs:element name="usage" type="TrafficAreaUsageType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="function" type="TrafficAreaFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="surfaceMaterial" type="TrafficSurfaceMaterialType" 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="function" type="AuxiliaryTrafficAreaFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="surfaceMaterial" type="TrafficSurfaceMaterialType" 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"/>
```

10.5.4 External code lists

The transportation model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.6):

- TransportationComplexFunctionType
- TransportationComplexUsageType
- TrafficAreaFunctionType
- TrafficAreaUsageType
- AuxiliaryTrafficAreaFunctionType
- TrafficSurfaceMaterialType

10.5.5 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 *TransportationComplex* 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.

10.6 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. 38). Single vegetation objects are modelled by the class *SolitaryVegetationObject*, while 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. 39, for XML schema definition see below and annex A.10.

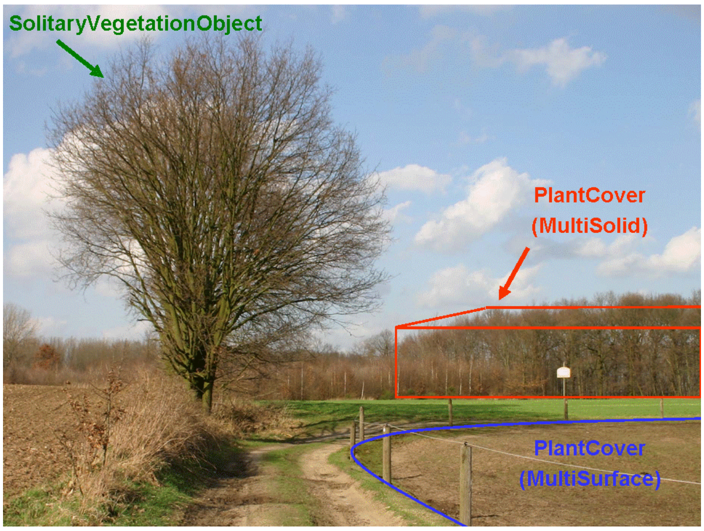


Fig. 38: Example for vegetation objects of the classes *SolitaryVegetationObject* and *PlantCover* (graphic: District of Recklinghausen).

A *SolitaryVegetationObject* may have the attributes *class*, *species, function, height*, *trunkDiameter* and *crownDiameter*. The attribute *class* contains the coarse classification of the object or plant habit, e.g. tree, bush, grass, and can occur only once (see external code list in chapter 10.6.4 and annex C.5). The attribute *species* defines the species' name, for example "Abies alba", and can occur at most once (see external code list in chapter 10.6.4 and annex C.5). The hierarchy between *class* and *species* is not reflected in the external code lists, thus inconsistencies have to be checked by application tools. The optional attribute *function* denotes the purpose of the object, for example botanical monument, and can occur multiple times. 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* and *averageHeight*. The plant community of a *PlantCover* is represented by the attribute *class.* The values of this attribute are enumerated in an external code list (cf. chapter 10.6.4 and annex C.5), where each value describes not only one plant type or species, but denotes 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 attribute *function* indicates the 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 *CityObjects*, 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).

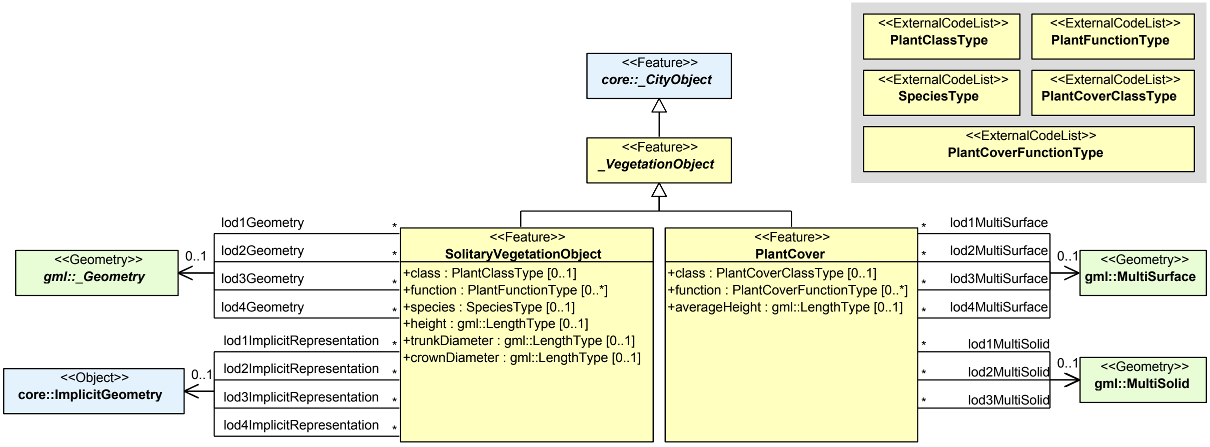


Fig. 39: UML diagram of vegetation objects in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Vegetation* module.

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* (see chapter 8.2). Solitary vegetation objects probably are one of the most important features where implicit geometries are appropriate, since the shape of the 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 *ImplicitGeometries.* For visualisation purposes, only the content of the library object defining the object's shape and appearance has to be swapped (cf. Fig. 40).

Fig. 40: 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 *Solitary-VegetationObject* is associated with the *_Geometry* class representing an arbitrary GML geometry (by the relation *lodXGeometry*, X ∈ [1..4]), a *PlantCover* is restricted to be either a *MultiSolid* or a *MultiSurface.* An example of a *PlantCover* modelled as *MultiSolid* is a '*solid forest model'*, see Fig. 41.

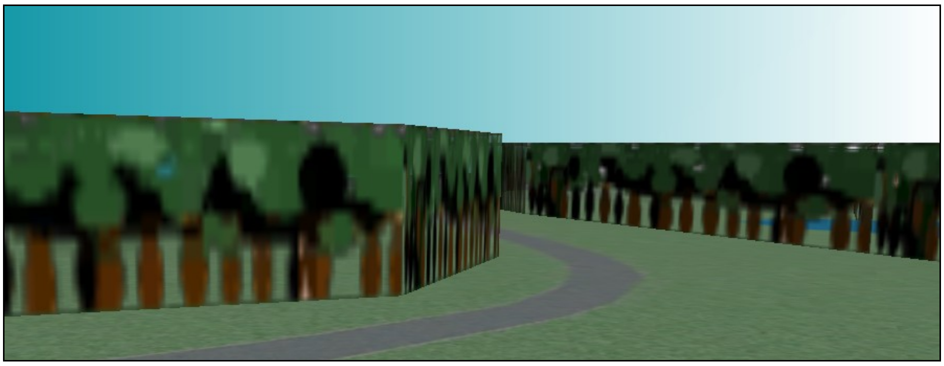


Fig. 41: Example for the visualisation/modelling of a solid forest (source: District of Recklinghausen).

XML namespace

The XML namespace of the CityGML *Vegetation* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/vegetation/1.0. Within the XML Schema definition of the *Vegetation* module, this URI is also used to identify the default namespace.

10.6.1 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" substitutionGroup="core:_CityObject"/>
<!-- ====================================================================== -->
<xs:element name="_GenericApplicationPropertyOfVegetationObject" type="xs:anyType" abstract="true"/>
```

10.6.2 Solitary vegetation objects

SolitaryVegetationObjectType, SolitaryVegetationObject

```
<xs:complexType name="SolitaryVegetationObjectType">
  <xs:complexContent>
    <xs:extension base="AbstractVegetationObjectType">
      <xs:sequence>
        <xs:element name="class" type="PlantClassType" minOccurs="0"/>
        <xs:element name="function" type="PlantFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="species" type="SpeciesType" 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"/>
```

10.6.3 Plant cover objects

PlantCoverType, PlantCover

```
<xs:complexType name="PlantCoverType">
  <xs:complexContent>
    <xs:extension base="AbstractVegetationObjectType">
      <xs:sequence>
        <xs:element name="class" type="PlantCoverClassType" minOccurs="0"/>
        <xs:element name="function" type="PlantCoverFunctionType" 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 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"/>
```

10.6.4 External code lists

The vegetation model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.5):

- PlantClassType
- PlantFunctionType
- SpeciesType
- PlantCoverClassType
- PlantCoverFunctionType

10.6.5 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.

```
<SolitaryVegetationObject>
  <class>1070</class>
  <species>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>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>
```

10.6.6 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.

10.7 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. 42, Fig. 43). 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. Or city furniture objects are important to plan a heavy vehicle transportation, where the exact position and further conditions of obstacles must be known.

Fig. 42: 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. 43: Real situation showing lanterns and delimitation stakes (left). In the right image they are modelled as *CityFurniture* objects with *ImplicitGeometries* (source: 3D city model of Barkenberg).

The UML diagram of the city furniture model is depicted in Fig. 44, for XML schema definition see below and annex A.4.

The class *CityFurniture* may have the attributes *class* and *function*. Their possible values are specified in the respective external code lists (chapter 10.7.2 and annex C.2). 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 (e.g. transportation, traffic regulation, architecture etc.), and can occur multiple times. The hierarchy between *class* and *function* is not reflected in the external code lists. Inconsistencies have to be checked by the application tools.

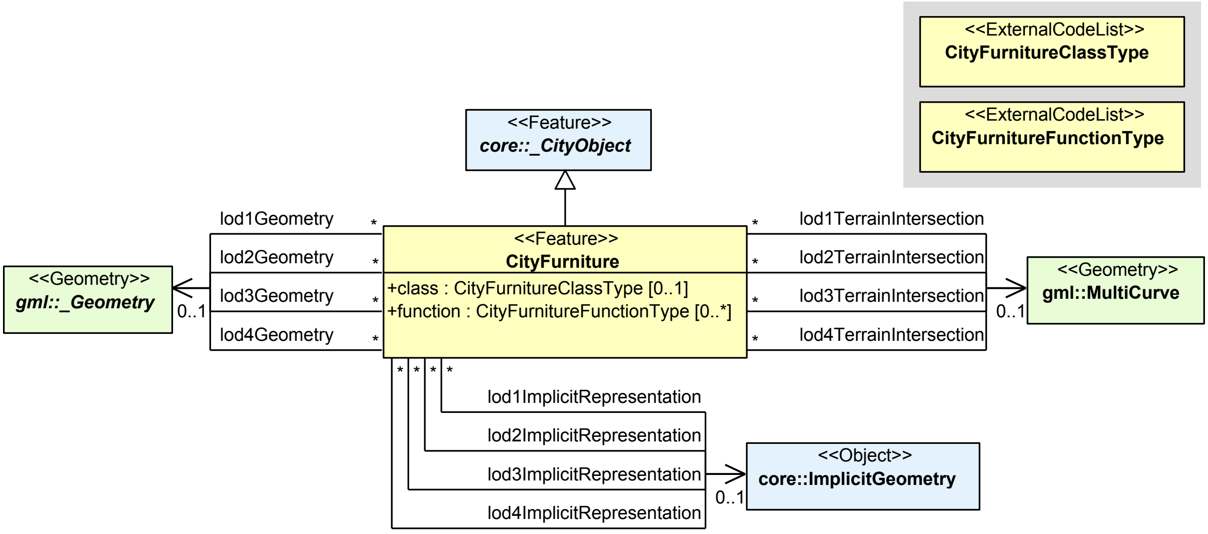


Fig. 44: UML diagram of city furniture objects in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *CityFurniture* module.

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 *GenericAttributes* using CityGML's *Generics* module (cf. chapter 10.10). For *ExternalReferences* city furniture objects can have links to external thematic databases. Thereby, semantical information of the objects, which can not 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). In the concept of *ImplicitGeometry* the geometry of a prototype city furniture object is stored only once in a local coordinate system and referenced by a number of features (see chapter 8.2). Spatial information of city furniture objects can be taken from city maps (called "Stadtgrundkarte" in Germany) 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/1.0. Within the XML Schema definition of the *CityFurniture* module, this URI is also used to identify the default namespace.

10.7.1 City furniture object

CityFurnitureType, CityFurniture

```
<xs:complexType name="CityFurnitureType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="CityFurnitureClassType" minOccurs="0"/>
        <xs:element name="function" type="CityFurnitureFunctionType" 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"/>
```

10.7.2 External code lists

The city furniture model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.2):

- CityFurnitureFunctionType
- CityFurnitureClassType

10.7.3 Example CityGML dataset

The following example of a CityGML dataset is an extract of the modelling of a delimitation stake in LOD3 and contains the attributes: *class* = 1000, *function* = 1520 (delimitation stake). The delimitation stake with the object ID *stake0815* is referencing by *urn:adv:oid:DEHE123400007001* to an cadastre object in the German ALKIS database (www.adv-online.de).

This example shows the geometry of *Cover Surface* (on the top of the stake) and of the left *Surface left* (Fig. 45). The *Cover Surface* has the material (color) white and the *Surface left* has the texture *stake.gif* with the relevant texture coordinates.

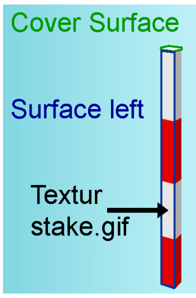


Fig. 45: Example of a simple city furniture object (source: District of Recklinghausen).

```
<!-- 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>1000</class>
  <!-- 1000 = traffic  -->
  <function>1520</function>
  <!-- 1520 = delimitation stake  -->
  <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>
```

10.7.4 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.

10.8 Land use

LandUse objects describe areas of the earth's surface dedicated to a specific land use. They can be employed to represent parcels in 3D. Fig. 46 shows the UML diagram of land use objects, for the XML schema definition see chapter 10.8.1 and annex A.7. The land use model of CityGML is provided by the thematic extension module *LandUse* (cf. chapter 7).

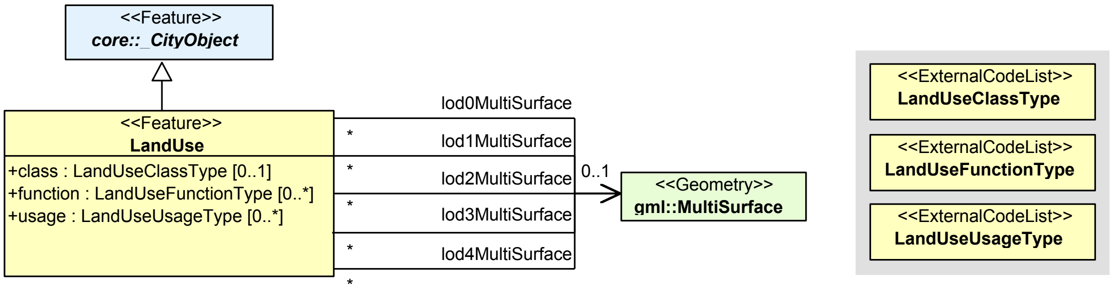


Fig. 46: UML diagram of land use objects in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *LandUse* module.

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 are specified in an external code list (see annex C.3). The attribute *function* defines the purpose of the object, like e.g. cornfield, 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. 47 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.

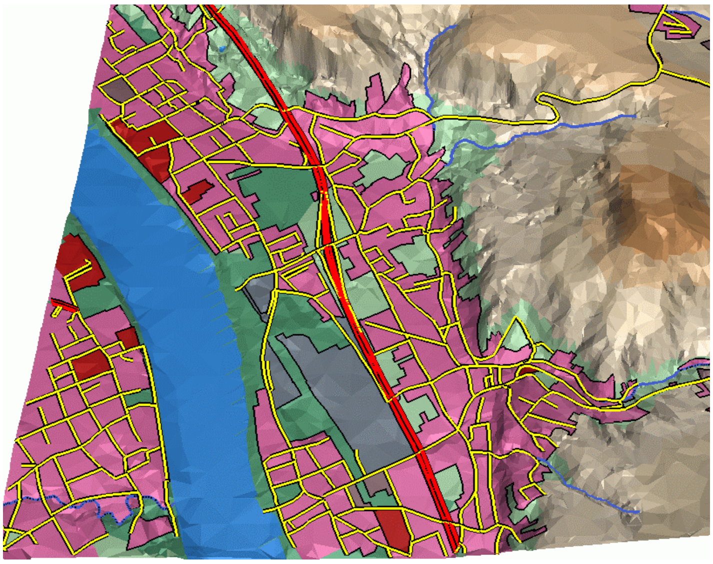


Fig. 47: LOD0 regional model consisting of land use objects in CityGML (source: IGG Uni Bonn).

XML namespace

The XML namespace of the CityGML *LandUse* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/landuse/1.0. Within the XML Schema definition of the *LandUse* module, this URI is also used to identify the default namespace.

10.8.1 Land use object

LandUseType, LandUse

```
<xs:complexType name="LandUseType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="LandUseClassType" minOccurs="0"/>
        <xs:element name="function" type="LandUseFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="LandUseUsageType" 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"/>
```

10.8.2 External code lists

The land use model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.3):

- LandUseClassType
- LandUseFunctionType
- LandUseUsageType

10.8.3 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.

10.9 City object groups

The grouping concept has already 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. 48 shows the UML diagram for the class *CityObjectGroup,* for the XML schema see annex A.5. The grouping concept of CityGML is defined by the thematic extension module *CityObjectGroup* (cf. chapter 7).

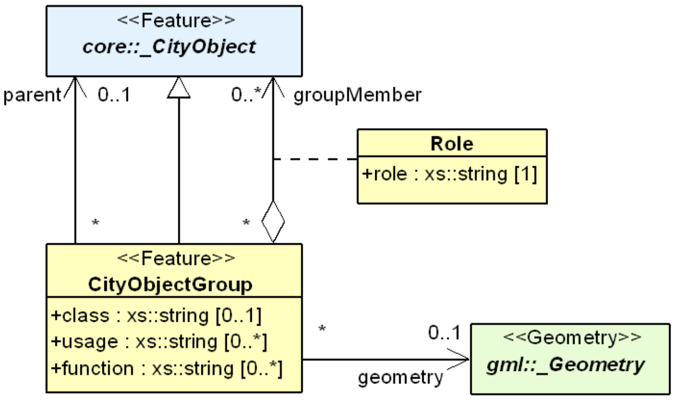


Fig. 48: UML diagram of city object groups in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *CityObjectGroup* module.

The class *CityObjectGroup* has the optional attributes *class, function* and *usage.* In contrast to the other thematic classes, no code lists are defined for these attributes, because the reasons for groupings can not completely be foreseen. 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. 8 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/1.0. Within the XML Schema definition of the *CityObjectGroup* module, this URI is also used to identify the default namespace.

10.9.1 City object group

CityObjectGroupType, CityObjectGroup

```
<xs:complexType name="CityObjectGroupType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
```



```
        <xs:element name="class" type="xs:string" minOccurs="0"/>
        <xs:element name="function" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="xs:string" 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>
```

10.9.2 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.

10.10 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. 49 shows the UML diagram of generic objects and attributes. For XML schema definition see below and annex A.6.

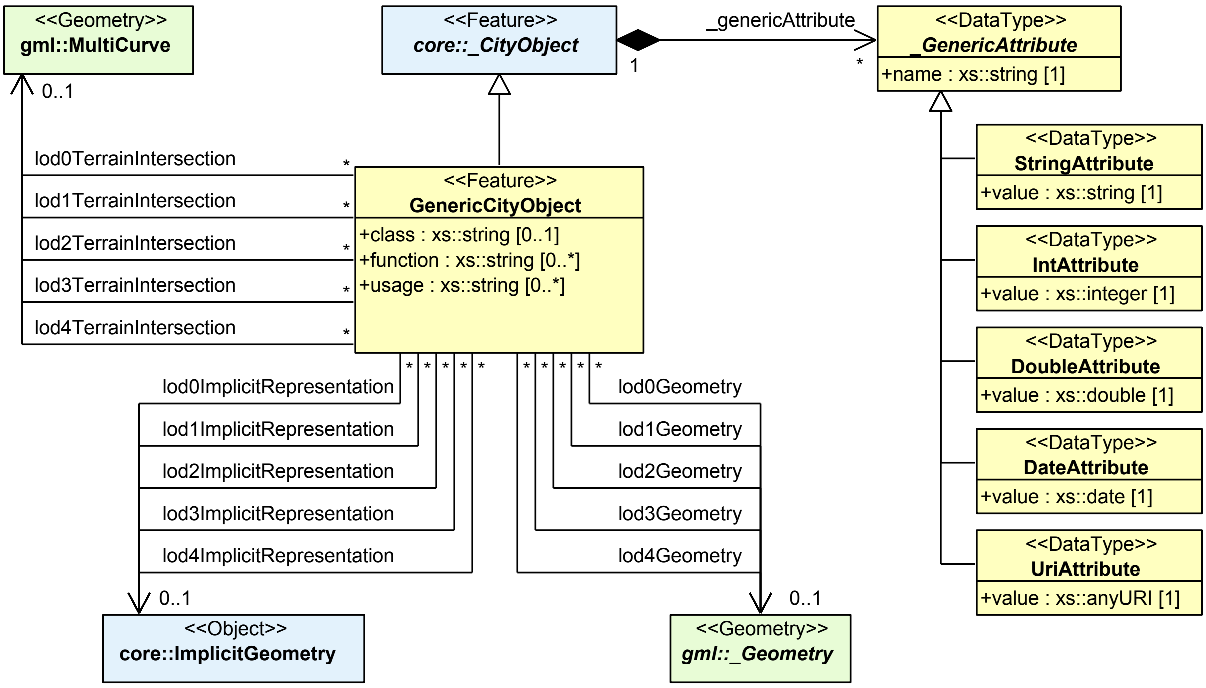


Fig. 49: UML diagram of generic objects and attributes in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Generics* module.

A *GenericCityObject* may have the attributes *class, function* and *usage* defined as *string*. The *class* attribute allows an object classification within the thematic area such as bridge, tunnel, 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.

In order to represent 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 subclasses of *_CityObject*. Data types of generic attributes may be *String*, *Integer*, *Double* (floating point number), *URI* and *Date*. The attribute type is defined by the selection of the particular subclass of *_GenericAttribute*, for example *StringAttribute*, *IntAttribute*, etc.

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 *_Geometry*, which is the base class of all GML geometries, *lodXGeometry,* X $\in$ [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 bridge should be represented as a *GenericCityObject,* a smooth transition between the DTM and the road on the bridge would have to be ensured (in order to avoid unrealistic bumps).

XML namespace

The XML namespace of the CityGML *Generics* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/generics/1.0. Within the XML Schema definition of the *Generics* module, this URI is also used to identify the default namespace.

10.10.1 Generic city object

GenericCityObjectType, GenericCityObject

```
<xs:complexType name="GenericCityObjectType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="xs:string" minOccurs="0"/>
        <xs:element name="function" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="xs:string" 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"/>
```

10.10.2 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"/>
```

10.10.3 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.

10.11 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 been already introduced as a first possibility to support the exchange of application specific data (see section 10.10). 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).

10.11.1 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* (see section 10.8.1).

   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 dataype 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).

10.11.2 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*). 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 G):

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns="http://www.citygml.org/ade/noise_de"
   xmlns:core="http://www.opengis.net/citygml/1.0"
   xmlns:bldg="http://www.opengis.net/citygml/building/1.0"
   xmlns:frn="http://www.opengis.net/citygml/cityfurniture/1.0"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:gml="http://www.opengis.net/gml" targetNamespace="http://www.citygml.org/ade/noise_de" elementFormDefault="qualified"
   attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
   <xsd:import namespace="http://www.opengis.net/citygml/1.0"
    schemaLocation="http://www.citygml.org/citygml/1.0/cityGMLBase.xsd"/>
   <xsd:import namespace="http://www.opengis.net/citygml/building/1.0"
    schemaLocation="http://www.citygml.org/citygml/building/1.0/building.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/cityfurniture/1.0"
    schemaLocation=" http://www.citygml.org/citygml/cityfurniture/1.0/cityfurniture.xsd"/>
   …
   <xsd:element name="buildingReflectionCorrection" type="gml:MeasureType"
      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.

```xml
…
  <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>
…
```


10.12 Definition of code lists

For the representation of city object attributes having an enumerative range of values, the concept of simple dictionaries is used, which is provided as a profile based on GML (cf. chapter 6.6). For each such attribute, the values are defined in a single file, which is named according to the attribute name. These files are located in a folder named *Codelists*, which comes with the CityGML schema document, but is not a normative part of this schema, since it may be modified, augmented, or replaced by other communities. The actual values in the files in the folder *Codelists* are a suggestion of the SIG 3D.

The external code list files define attribute values and assign an unique identifier to each value. In a CityGML instance document, an attribute value is denoted by an identifier of a value, not by the value itself. Thus typos are avoided and it is ensured that the same concept is denoted the same way, by the same identifier and not by two different terms with identical meaning. Thus the use of code lists facilitates semantic and syntactic interoperability, since they define common terms within an information community. Furthermore, the simple dictionary concept enables more than one term to be assigned to the same dictionary entry, thus the same concept may be explained in different languages. To differentiate between the languages, code spaces may be used.

An example for an enumerative attribute is *RoofType*, which is defined by the following excerpt of the external code list file:

```
<gml:Dictionary xmlns="http://www.opengis.net/gml" xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/gml ../3.1.1/profiles/simpleDictionary/1.0.0/gmlSimpleDictionaryProfile.xsd"
  gml:id="RoofTypeType">
  <name>RoofTypeType</name>
  <dictionaryEntry>
    <gml:Definition gml:id="id357">
      <gml:description/>
      <gml:name codeSpace="urn:d_nrw_sig3d">1000</gml:name>
      <gml:name>flat roof</gml:name>
    </gml:Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <gml:Definition gml:id="id358">
      <gml:description/>
      <gml:name codeSpace="urn:d_nrw_sig3d">1010</gml:name>
      <gml:name>monopitch roof</gml:name>
    </gml:Definition>
  </dictionaryEntry>
  ….
</gml:Dictionary>
```

In the simple dictionary concept, the values of an attribute are represented by a *Dictionary* element, where each value is given by a *dictionaryEntry* resp. *Definition* entry. In CityGML, a definition entry is identified by the *name* element, which is qualified by the SIG 3D code space. The unqualified *name* element represents the value of the attribute. An optional description explains the value. CityGML does not use GML identifiers (*gml:id*) to link to attribute values, since IDs are restricted syntactically, and must be globally unique, which is not feasible for code lists.

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/1.0 is associated with the core module.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/1.0" 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/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:import namespace="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0" schemaLocation="../xAL/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 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>
```



```xml
      <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: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:complexContent>
      <xs:restriction base="gml:AssociationType">
        <xs:sequence minOccurs="0">
          <xs:element ref="Address"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </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="MimeTypeType" 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:complexContent>
      <xs:restriction base="gml:AssociationType">
        <xs:sequence minOccurs="0">
          <xs:element ref="ImplicitGeometry"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </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:simpleType name="MimeTypeType">
    <xs:annotation>
      <xs:documentation>MIME type of a geometry in an external library file. MIME types are defined by the IETF (Internet
        Engineering Task Force). The values of this type are defined in the XML file MimeTypeType.xml, according to the
        dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </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/1.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/appearance/1.0" xmlns:core="http://www.opengis.net/citygml/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/citygml/appearance/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="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="_GenericApplicationPropertyOfAppearance" type="xs:anyType" abstract="true"/>
  <!-- ========================================================================================= -->
  <xs:complexType name="AppearancePropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of a _CityObject resp. CityModel 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:complexContent>
      <xs:extension base="gml:FeaturePropertyType">
        <xs:sequence minOccurs="0">
          <xs:element name="Appearance" type="AppearanceType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================= -->
  <xs:element name="appearanceMember" type="AppearancePropertyType" substitutionGroup="gml:featureMember"/>
  <!-- ========================================================================================= -->
  <xs:element name="appearance" type="AppearancePropertyType" substitutionGroup="core:_GenericApplicationPropertyOfCityObject"/>
  <!-- ========================================================================================= -->
  <xs:complexType name="AbstractSurfaceDataType">
    <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">
    <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="core:MimeTypeType" 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>
```



```xml
<!-- ========================================================================================= -->
<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 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/1.0 is associated with this extension module.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/building/1.0" xmlns:core="http://www.opengis.net/citygml/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/citygml/building/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="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="BuildingClassType" minOccurs="0"/>
          <xs:element name="function" type="BuildingFunctionType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="BuildingUsageType" 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="RoofTypeType" 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="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>
```



```xml
<!-- ========================================================================================== -->
<xs:element name="_AbstractBuilding" type="AbstractBuildingType" abstract="true" substitutionGroup="core:_Site"/>
<!-- ========================================================================================== -->
<xs:element name="_GenericApplicationPropertyOfAbstractBuilding" type="xs:anyType" abstract="true"/>
<!-- ========================================================================================== -->
<xs:simpleType name="BuildingClassType">
  <xs:annotation>
    <xs:documentation> Class of a building. The values of this type are defined in the XML file BuildingClassType.xml,
       according to the dictionary concept of GML3.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ========================================================================================== -->
<xs:simpleType name="BuildingFunctionType">
  <xs:annotation>
    <xs:documentation> Intended function of a building. The values of this type are defined in the XML file
       BuildingFunctionType.xml, according to the dictionary concept of GML3. The values may be adopted from ALKIS, the
       german standard for cadastre modelling. If the cadastre models from other countries differ in the building
       functions, these values may be compiled in another codelist to be used with CityGML. </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ========================================================================================== -->
<xs:simpleType name="BuildingUsageType">
  <xs:annotation>
    <xs:documentation> Actual usage of a building. The values of this type are defined in the XML file
       BuildingUsageType.xml, according to the dictionary concept of GML3.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ========================================================================================== -->
<xs:simpleType name="RoofTypeType">
  <xs:annotation>
    <xs:documentation>Roof Types. The values of this type are defined in the XML file RoofTypeType.xml, according to the
       dictionary concept of GML3.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ========================================================================================== -->
<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:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
```



```xml
        <xs:element ref="BuildingPart"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</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="BuildingInstallationClassType" minOccurs="0"/>
        <xs:element name="function" type="BuildingInstallationFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="BuildingInstallationUsageType" 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 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:simpleType name="BuildingInstallationClassType">
  <xs:annotation>
    <xs:documentation>Class of a building installation. The values of this type are defined in the XML file
       BuildingInstallationClassType.xml, according to the dictionary concept of GML3. </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ====================================================================================== -->
<xs:simpleType name="BuildingInstallationFunctionType">
  <xs:annotation>
    <xs:documentation>Function of a building installation. The values of this type are defined in the XML file
       BuildingInstallationFunctionType.xml, according to the dictionary concept of GML3. </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ====================================================================================== -->
<xs:simpleType name="BuildingInstallationUsageType">
  <xs:annotation>
    <xs:documentation>Actual usage of a building installation. The values of this type are defined in the XML file
       BuildingInstallationUsageType.xml, according to the dictionary concept of GML3. </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ====================================================================================== -->
<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:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="BuildingInstallation"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</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="IntBuildingInstallationClassType" minOccurs="0"/>
          <xs:element name="function" type="IntBuildingInstallationFunctionType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="IntBuildingInstallationUsageType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
          <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:simpleType name="IntBuildingInstallationClassType">
    <xs:annotation>
      <xs:documentation>Class of an interior building installation. The values of this type are defined in the XML file
        IntBuildingInstallationClassType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================== -->
  <xs:simpleType name="IntBuildingInstallationFunctionType">
    <xs:annotation>
      <xs:documentation>Function of an interior building installation. The values of this type are defined in the XML file
        IntBuildingInstallationFunctionType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================== -->
  <xs:simpleType name="IntBuildingInstallationUsageType">
    <xs:annotation>
      <xs:documentation>Actual Usage of an interior building installation. The values of this type are defined in the XML
        file IntBuildingInstallationUsageType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================== -->
  <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:complexContent>
      <xs:restriction base="gml:AssociationType">
        <xs:sequence minOccurs="0">
          <xs:element ref="IntBuildingInstallation"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
  <!-- ========================================================================================== -->
  <xs:complexType name="AbstractBoundarySurfaceType" abstract="true">
    <xs:annotation>
      <xs:documentation>A BoundarySurface is a thematic object which classifies surfaces bounding a building or a room. 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>
```



```xml
</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"/>
<!-- ========================================================================================== -->
```



```xml
<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 _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 ClosureSurfaces belong 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:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="_BoundarySurface"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</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:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="_Opening"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
<!-- ========================================================================================= -->
<xs:complexType name="AbstractOpeningType" abstract="true">
  <xs:annotation>
    <xs:documentation> Type for openings (doors, windows) in walls. 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 ref="_GenericApplicationPropertyOfOpening" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
```



```xml
</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 walls. 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 walls. 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="RoomClassType" minOccurs="0"/>
        <xs:element name="function" type="RoomFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="RoomUsageType" 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"/>
<!-- ====================================================================== -->
```



```xml
<xs:simpleType name="RoomClassType">
  <xs:annotation>
    <xs:documentation>Class of a room . The values of this type are defined in the XML file RoomClassType.xml, according
        to the dictionary concept of GML3. </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ========================================================================================= -->
<xs:simpleType name="RoomFunctionType">
  <xs:annotation>
    <xs:documentation>Function of a room. The values of this type are defined in the XML file RoomFunctionType.xml,
        according to the dictionary concept of GML3. </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ========================================================================================= -->
<xs:simpleType name="RoomUsageType">
  <xs:annotation>
    <xs:documentation>Actual Usage of a room. The values of this type are defined in the XML file RoomUsageType.xml,
        according to the dictionary concept of GML3. </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ========================================================================================= -->
<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="BuildingFurnitureClassType" minOccurs="0"/>
        <xs:element name="function" type="BuildingFurnitureFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="BuildingFurnitureUsageType" 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:simpleType name="BuildingFurnitureClassType">
  <xs:annotation>
    <xs:documentation> Class of a building furniture. The values of this type are defined in the XML file
        BuildingFurnitureClassType.xml, according to the dictionary concept of GML3. </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ========================================================================================= -->
<xs:simpleType name="BuildingFurnitureFunctionType">
  <xs:annotation>
    <xs:documentation> Function of a building furniture. The values of this type are defined in the XML file
        BuildingFurnitureFunctionType.xml, according to the dictionary concept of GML3. </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ========================================================================================= -->
<xs:simpleType name="BuildingFurnitureUsageType">
  <xs:annotation>
    <xs:documentation> Actual Usage of a building Furniture. The values of this type are defined in the XML file
        BuildingFurnitureUsageType.xml, according to the dictionary concept of GML3. </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ========================================================================================= -->
<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:complexContent>
      <xs:restriction base="gml:AssociationType">
        <xs:sequence minOccurs="0">
          <xs:element ref="Room"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </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:complexContent>
      <xs:restriction base="gml:AssociationType">
        <xs:sequence minOccurs="0">
          <xs:element ref="BuildingFurniture"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

A.4 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/1.0 is associated with this extension module.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/cityfurniture/1.0" xmlns:core="http://www.opengis.net/citygml/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/citygml/cityfurniture/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="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="CityFurnitureClassType" minOccurs="0"/>
          <xs:element name="function" type="CityFurnitureFunctionType" 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:simpleType name="CityFurnitureFunctionType">
    <xs:annotation>
      <xs:documentation>Function of a Furniture. The values of this type are defined in the XML file
          CityFurnitureFunctionType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================== -->
  <xs:simpleType name="CityFurnitureClassType">
    <xs:annotation>
      <xs:documentation>Class of a Furniture. The values of this type are defined in the XML file
          CityFurnitureClassType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
</xs:schema>
```


A.5 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/1.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/cityobjectgroup/1.0" xmlns:core="http://www.opengis.net/citygml/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/citygml/cityobjectgroup/1.0" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="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="xs:string" minOccurs="0"/>
          <xs:element name="function" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="xs:string" 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.6 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/1.0 is associated with this extension module.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/generics/1.0" xmlns:core="http://www.opengis.net/citygml/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/citygml/generics/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="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. Otherwise, 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="xs:string" minOccurs="0"/>
          <xs:element name="function" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="xs:string" 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>
```



```xml
      </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:schema>
```


A.7 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/1.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/landuse/1.0" xmlns:core="http://www.opengis.net/citygml/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/citygml/landuse/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="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="LandUseClassType" minOccurs="0"/>
          <xs:element name="function" type="LandUseFunctionType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="LandUseUsageType" 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:simpleType name="LandUseClassType">
    <xs:annotation>
      <xs:documentation>Class of a Landuse. The values of this type are defined in the XML file LandUseClassType.xml,
          according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================= -->
  <xs:simpleType name="LandUseFunctionType">
    <xs:annotation>
      <xs:documentation>Function of a Landuse. The values of this type are defined in the XML file LandUseFunctionType.xml,
          according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================= -->
  <xs:simpleType name="LandUseUsageType">
    <xs:annotation>
      <xs:documentation>Usage of a Landuse. The values of this type are defined in the XML file LandUseUsageType.xml,
          according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
</xs:schema>
```


A.8 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/1.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/relief/1.0" xmlns:core="http://www.opengis.net/citygml/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/citygml/relief/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="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:complexContent>
      <xs:restriction base="gml:AssociationType">
        <xs:sequence minOccurs="0">
          <xs:element ref="_ReliefComponent"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </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:complexContent>
      <xs:restriction base="gml:AssociationType">
        <xs:sequence minOccurs="0">
          <xs:element ref="gml:TriangulatedSurface"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
  <!-- ============================================================================== -->
  <xs:complexType name="gridPropertyType">
    <xs:complexContent>
      <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:restriction base="gml:AssociationType">
        <xs:sequence minOccurs="0">
          <xs:element ref="gml:RectifiedGridCoverage"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
  <!-- ============================================================================== -->
  <xs:element name="Elevation" type="gml:LengthType" substitutionGroup="gml:_Object"/>
</xs:schema>
```


A.9 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/1.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/transportation/1.0" xmlns:core="http://www.opengis.net/citygml/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/citygml/transportation/1.0" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="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" 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="function" type="TransportationComplexFunctionType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="TransportationComplexUsageType" 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="usage" type="TrafficAreaUsageType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="function" type="TrafficAreaFunctionType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="surfaceMaterial" type="TrafficSurfaceMaterialType" 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="function" type="AuxiliaryTrafficAreaFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="surfaceMaterial" type="TrafficSurfaceMaterialType" 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:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="TrafficArea"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</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:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="AuxiliaryTrafficArea"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</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:simpleType name="TransportationComplexFunctionType">
  <xs:annotation>
    <xs:documentation>Function of a transportation complex. The values of this type are defined in the XML file
       TransportationComplexFunctionType.xml, according to the dictionary concept of GML3. </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- ========================================================================================== -->
<xs:simpleType name="TransportationComplexUsageType">
  <xs:annotation>
    <xs:documentation>Actual Usage of a transportation complex. The values of this type are defined in the XML file
```



```
          TransportationComplexUsageType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================= -->
  <xs:simpleType name="TrafficAreaFunctionType">
    <xs:annotation>
      <xs:documentation>Function of a traffic area. The values of this type are defined in the XML file
          TrafficAreaFunctionType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================= -->
  <xs:simpleType name="AuxiliaryTrafficAreaFunctionType">
    <xs:annotation>
      <xs:documentation>Function of an auxiliary traffic area. The values of this type are defined in the XML file
          AuxiliaryTrafficAreaFunctionType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================= -->
  <xs:simpleType name="TrafficAreaUsageType">
    <xs:annotation>
      <xs:documentation>Usage of a traffic area. The values of this type are defined in the XML file
          TrafficAreaUsageType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================= -->
  <xs:simpleType name="TrafficSurfaceMaterialType">
    <xs:annotation>
      <xs:documentation>Type for surface materials of transportation objects. The values of this type are defined in the XML
          file TrafficSurfaceMaterialType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
</xs:schema>
```


A.10 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/1.0 is associated with this extension module.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/vegetation/1.0" xmlns:core="http://www.opengis.net/citygml/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/citygml/vegetation/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="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" 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="PlantCoverClassType" minOccurs="0"/>
          <xs:element name="function" type="PlantCoverFunctionType" 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 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:simpleType name="PlantCoverClassType">
    <xs:annotation>
      <xs:documentation>Class of a PlantCover. The values of this type are defined in the XML file PlantCoverClassType.xml,
          according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================= -->
  <xs:simpleType name="PlantCoverFunctionType">
    <xs:annotation>
      <xs:documentation>Function of a PlantCover. The values of this type are defined in the XML file
          PlantCoverFunctionType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
```



```xml
  </xs:simpleType>
  <!-- ============================================================================================ -->
  <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="PlantClassType" minOccurs="0"/>
          <xs:element name="function" type="PlantFunctionType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="species" type="SpeciesType" 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:simpleType name="PlantClassType">
    <xs:annotation>
      <xs:documentation>Class of a SolitaryVegetationObject. The values of this type are defined in the XML file
          PlantClassType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ============================================================================================ -->
  <xs:simpleType name="PlantFunctionType">
    <xs:annotation>
      <xs:documentation>Function of a PlantType. The values of this type are defined in the XML file PlantFunctionType.xml,
          according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ============================================================================================ -->
  <xs:simpleType name="SpeciesType">
    <xs:annotation>
      <xs:documentation>Type of a Species. The values of this type are defined in the XML file SpeciesType.xml, according to
          the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
</xs:schema>
```


A.11 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/1.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/waterbody/1.0" xmlns:core="http://www.opengis.net/citygml/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/citygml/waterbody/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="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" 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="WaterBodyClassType" minOccurs="0"/>
          <xs:element name="function" type="WaterBodyFunctionType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="WaterBodyUsageType" 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:simpleType name="WaterBodyClassType">
    <xs:annotation>
      <xs:documentation>Class of a Water Body. The values of this type are defined in the XML file WaterBodyClassType.xml,
         according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================= -->
  <xs:simpleType name="WaterBodyFunctionType">
    <xs:annotation>
      <xs:documentation>Function of a Water Body. The values of this type are defined in the XML file
```



```xml
            WaterBodyFunctionType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================== -->
  <xs:simpleType name="WaterBodyUsageType">
    <xs:annotation>
      <xs:documentation>Actual usage of a water body. The values of this type are defined in the XML file
            WaterBodyUsageType.xml, according to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================== -->
  <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:complexContent>
      <xs:restriction base="gml:AssociationType">
        <xs:sequence minOccurs="0">
          <xs:element ref="_WaterBoundarySurface"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </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:simpleType name="WaterLevelType">
    <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:restriction base="xs:string"/>
  </xs:simpleType>
  <!-- ========================================================================================== -->
  <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="WaterLevelType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfWaterSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```



```xml
  <!-- ========================================================================================= -->
  <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.12 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/1.0 is associated with this extension module.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/texturedsurface/1.0" xmlns:core="http://www.opengis.net/citygml/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/citygml/texturedsurface/1.0" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="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.13 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 10.3.8 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 1.0 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"?>
<schema xmlns="http://www.ascc.net/xml/schematron">
  <title>Schematron validation rules checking referential integrity of CityGML association types</title>
  <ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>
  <ns prefix="bldg" uri="http://www.opengis.net/citygml/building/1.0"/>
  …
  <pattern name="Check either href or content not both">
    <rule abstract="true" id="hrefOrContent">
      <report test="@xlink:href and (*|text())">Property element may not carry both a reference to an object and contain an
        object.</report>
      <assert test="@xlink:href | (*|text())">Property element must either carry a reference to an object or contain an
        object.</assert>
    </rule>
  </pattern>
  …
</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.

```
  …
  <pattern name="hrefOrContent check on bldg:consistsOfBuildingPart">
    <rule context="bldg:consistsOfBuildingPart">
      <extends rule="hrefOrContent"/>
    </rule>
  </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

B.1.2.1 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.

B.1.2.2 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

B.2.1.1 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 0 and 8.2.3. 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.13.
c) Reference	Chapter 10.1, 0, 8.2, 8.2.3, annex A.13.
d) Test type	Basic Test.

B.2.1.2 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

B.2.2.1 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.13.
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 9.6.
c) Reference	Chapter 9, 9.6, annex A.13.
d) Test type	Capability Test.

B.2.2.2 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 Building module

B.2.3.1 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.13.
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 10.3.8.
c) Reference	Chapter 10.3, 10.3.8, annex A.13.
d) Test type	Capability Test.

B.2.3.2 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.3.
d) Test type	Capability Test.

B.2.4 CityFurniture module

B.2.4.1 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.13.
b) Test method	Inspect the instance document and check that it satifies the rules of the *CityFurniture* module described in chapter 10.7, and especially the conformance requirements stated in chapter 10.7.4.
c) Reference	Chapter 10.7, 10.7.4, annex A.13.

d) Test type	Capability Test.

B.2.4.2 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.4. 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.4.
d) Test type	Capability Test.

B.2.5 CityObjectGroup module

B.2.5.1 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.13.
b) Test method	Inspect the instance document and check that it satifies the rules of the *CityObjectGroup* module described in chapter 10.9, and especially the conformance requirements stated in chapter 10.9.2.
c) Reference	Chapter 10.9, 10.9.2, annex A.13.
d) Test type	Capability Test.

B.2.5.2 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.5. 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.5.
d) Test type	Capability Test.

B.2.6 Generics module

B.2.6.1 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.13.
b) Test method	Inspect the instance document and check that it satifies the rules of the *Generics* module described in chapter 10.10, and especially the conformance requirements stated in chapter 10.10.2.
c) Reference	Chapter 10.10, 10.10.2, annex A.13.
d) Test type	Capability Test.

B.2.6.2 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.6. 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.6.
d) Test type	Capability Test.

B.2.7 LandUse module

B.2.7.1 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.13.
b) Test method	Inspect the instance document and check that it satifies the rules of the *LandUse* module described in chapter 10.8, and especially the conformance requirements stated in chapter 10.8.3.
c) Reference	Chapter 10.8, 10.8.3, annex A.13.
d) Test type	Capability Test.

B.2.7.2 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.7. 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.7.
d) Test type	Capability Test.

B.2.8 Relief module

B.2.8.1 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.13.
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.13.
d) Test type	Capability Test.

B.2.8.2 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.8. 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.8.
d) Test type	Capability Test.

B.2.9 Transportation module

B.2.9.1 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.13.
b) Test method	Inspect the instance document and check that it satifies the rules of the *Transportation* module described in chapter 10.5, and especially the conformance requirements stated in chapter 10.5.5.
c) Reference	Chapter 10.5, 10.5.5, annex A.13.
d) Test type	Capability Test.

B.2.9.2 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.9. 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.9.
d) Test type	Capability Test.

B.2.10 Vegetation module

B.2.10.1 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.13.
b) Test method	Inspect the instance document and check that it satifies the rules of the *Vegetation* module described in chapter 10.6, and especially the conformance requirements stated in chapter 10.6.6.
c) Reference	Chapter 10.6, 10.6.6, annex A.13.
d) Test type	Capability Test.

B.2.10.2 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.10. 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.10.
d) Test type	Capability Test.

B.2.11 WaterBody module

B.2.11.1 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.13.
b) Test method	Inspect the instance document and check that it satifies the rules of the *WaterBody* module described in chapter 10.4, and especially the conformance requirements stated in chapter 10.4.4.
c) Reference	Chapter 10.4, 10.4.4, annex A.13.
d) Test type	Capability Test.

B.2.11.2 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.11. 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.11.
d) Test type	Capability Test.

B.2.12 TexturedSurface module [deprecated]

B.2.12.1 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.13.
b) Test method	Inspect the instance document and check that it satifies the rules of the *TexturedSurface* module described in chapter 9.7, and especially the conformance requirements stated in chapter 9.7.2.
c) Reference	Chapter 9.7, 9.7.2, annex A.13.
d) Test type	Capability Test.

B.2.12.2 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.12. 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.12.
d) Test type	Capability Test.

Annex C
(informative)

External code lists

In this annex the corresponding values of the external code lists of CityGML (cf. chapter 6.6) are given. The following values are a proposal of the SIG 3D and may be extended or replaced by other communities to fit their needs. The external code list for roof types is given in XML format 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 fields and comprise the relevant external code lists of these fields.

External code list for roof types, which are defined in the file RoofTypeType.xml:

```
<gml:Dictionary xmlns="http://www.opengis.net/gml" xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/gml ../3.1.1/profiles/simpleDictionary/1.0.0/gmlSimpleDictionaryProfile.xsd"
  gml:id="RoofTypeType">
  <name>RoofTypeType</name>
  <dictionaryEntry>
    <gml:Definition gml:id="id357">
      <gml:description/>
      <gml:name codeSpace="urn:d_nrw_sig3d">1000</gml:name>
      <gml:name>flat roof</gml:name>
    </gml:Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <gml:Definition gml:id="id358">
      <gml:description/>
      <gml:name codeSpace="urn:d_nrw_sig3d">1010</gml:name>
      <gml:name>monopitch roof</gml:name>
    </gml:Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <gml:Definition gml:id="id359">
      <gml:description/>
      <gml:name codeSpace="urn:d_nrw_sig3d">1020</gml:name>
      <gml:name>skip pent roof</gml:name>
    </gml:Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <gml:Definition gml:id="id360">
      <gml:description/>
      <gml:name codeSpace="urn:d_nrw_sig3d">1030</gml:name>
      <gml:name>gabled roof</gml:name>
    </gml:Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <gml:Definition gml:id="id361">
      <gml:description/>
      <gml:name codeSpace="urn:d_nrw_sig3d">1040</gml:name>
      <gml:name>hipped roof</gml:name>
    </gml:Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <gml:Definition gml:id="id362">
      <gml:description/>
      <gml:name codeSpace="urn:d_nrw_sig3d">1050</gml:name>
      <gml:name>half-hipped roof</gml:name>
    </gml:Definition>
  </dictionaryEntry>
  <dictionaryEntry>
    <gml:Definition gml:id="id363">
      <gml:description/>
      <gml:name codeSpace="urn:d_nrw_sig3d">1060</gml:name>
      <gml:name>mansard roof</gml:name>
    </gml:Definition>
  </dictionaryEntry>
  <dictionaryEntry>
```



```xml
      <gml:Definition gml:id="id364">
        <gml:description/>
        <gml:name codeSpace="urn:d_nrw_sig3d">1070</gml:name>
        <gml:name>pavilion roof</gml:name>
      </gml:Definition>
    </dictionaryEntry>
    <dictionaryEntry>
      <gml:Definition gml:id="id365">
        <gml:description/>
        <gml:name codeSpace="urn:d_nrw_sig3d">1080</gml:name>
        <gml:name>cone roof</gml:name>
      </gml:Definition>
    </dictionaryEntry>
    <dictionaryEntry>
      <gml:Definition gml:id="id366">
        <gml:description/>
        <gml:name codeSpace="urn:d_nrw_sig3d">1090</gml:name>
        <gml:name>copula roof</gml:name>
      </gml:Definition>
    </dictionaryEntry>
    <dictionaryEntry>
      <gml:Definition gml:id="id367">
        <gml:description/>
        <gml:name codeSpace="urn:d_nrw_sig3d">1100</gml:name>
        <gml:name>shed roof</gml:name>
      </gml:Definition>
    </dictionaryEntry>
    <dictionaryEntry>
      <gml:Definition gml:id="id368">
        <gml:description/>
        <gml:name codeSpace="urn:d_nrw_sig3d">1110</gml:name>
        <gml:name>arch roof</gml:name>
      </gml:Definition>
    </dictionaryEntry>
    <dictionaryEntry>
      <gml:Definition gml:id="id369">
        <gml:description/>
        <gml:name codeSpace="urn:d_nrw_sig3d">1120</gml:name>
        <gml:name>pyramidal broach roof</gml:name>
      </gml:Definition>
    </dictionaryEntry>
    <dictionaryEntry>
      <gml:Definition gml:id="id370">
        <gml:description/>
        <gml:name codeSpace="urn:d_nrw_sig3d">1130</gml:name>
        <gml:name>combination of roof forms</gml:name>
      </gml:Definition>
    </dictionaryEntry>
</gml:Dictionary>
```

C.1 Building

BuildingClassType			
Code list derived from German authoritative standards ALKIS/ATKIS (www.adv-online.de)			
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		

BuildingFunctionType			
Code list derived from German authoritative standards ALKIS/ATKIS (www.adv-online.de)			
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 builiding	1900	equestrian hall
1070	agrarian- and forestry builiding	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		

BuildingUsageType
Code list identically specified as *BuildingFunctionType*

BuildingFurnitureClassType			
Code list proposed by the SIG 3D			
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		

BuildingFurnitureFunctionType			
Code list proposed by the SIG 3D			
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

BuildingFurnitureUsageType
Code list identically specified as *BuildingFurnitureFunctionType*

BuildingInstallationClassType			
Code list derived from German authoritative standards ALKIS/ATKIS (www.adv-online.de)			
1000	outer and inner characteristics	1040	communicating
1020	waste management	1050	security
1030	maintenance	1060	others

BuildingInstallationFunctionType			
Code list derived from German authoritative standards ALKIS/ATKIS (www.adv-online.de)			
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

BuildingInstallationUsageType
Code list identically specified as *BuildingInstallationFunctionType*

IntBuildingInstallationClassType			
Code list proposed by the SIG 3D			
1000	Heating, Ventilation, Climate	6000	Statics
2000	Safety	7000	Entertainmant
3000	Illumination	8000	Miscellaneous
4000	Communication	9999	Unknown
5000	Supply and Disposal		

IntBuildingInstallationFunctionType			
Code list proposed by the SIG 3D			
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

IntBuildingInstallationUsageType
Code list identically specified as *IntBuildingInstallationFunctionType*

RoofTypeType			
Code list derived from German authoritative standards ALKIS/ATKIS (www.adv-online.de)			
1000	flat roof	1070	pavilion roof
1010	monopitch roof	1080	cone roof
1020	skip pent roof	1090	copula roof
1030	gabled roof	1100	shed roof
1040	hipped roof	1110	arch roof
1050	half-hipped roof	1120	pyramidal broach roof
1060	mansard roof	1130	combination of roof forms

RoomClassType			
Code list proposed by the SIG 3D			
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

RoomFunctionType			
Code list proposed by the SIG 3D			
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

RoomUsageType
Code list identically specified as *RoomFunctionType*

C.2 City furniture

CityFurnitureClassType			
Code list derived from German authoritative standards ALKIS/ATKIS (www.adv-online.de)			
1000	traffic	1020	others
1010	communication		

CityFurnitureFunctionType			
Code list derived from German authoritative standards ALKIS/ATKIS (www.adv-online.de)			
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.3 Land use

LandUseClassType			
Code list derived from German authoritative Standard ATKIS (www.adv-online.de)			
1000	Settlement Area	3000	Vegetation
1100	Undeveloped Area	4000	Water
2000	Traffic		

LandUseFunctionType			
Code list proposed by the SIG 3D			
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		

LandUseUsageType			
Code list proposed by the SIG 3D			
1000	Civil	1300	Religious
1100	Private	1400	Military
1200	Public		

C.4 MimeType

MimeTypeType			
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.5 Vegetation

PlantClassType			
Code list proposed by the SIG 3D. It is based on information extracted from http://www.bundessortenamt.de and http://www.forst-hamburg.de/baumarten.htm.			
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

PlantFunctionType
Code list identically specified as *PlantClassType*

PlantCoverClassType			
Code list proposed by the SIG 3D. It is based on information extracted from http://www.biologie.uni-hamburg.de/b-online/e57/57.htm (Tab.3).			
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		

PlantCoverFunctionType
Code list identically specified as *PlantCoverClassType*

SpeciesType (excerpt)			
Code list proposed by the SIG 3D. It is based on information extracted from http://www.bundessortenamt.de and http://www.forst-hamburg.de/baumarten.htm.			
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.6 Transportation

AuxiliaryTrafficAreaFunctionType			
Code list proposed by the SIG 3D			
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		

TrafficAreaFunctionType			
Code list proposed by the SIG 3D			
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

TrafficAreaUsageType			
Code list proposed by the SIG 3D			
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		

TrafficSurfaceMaterialType			
Code list proposed by the SIG 3D			
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

TransportationComplexClass			
Code list proposed by the SIG 3D			
1000	private	1050	air traffic
1010	common	1060	rail traffic
1020	civil	1070	waterway
1030	military	1080	subway
1040	road traffic	1090	others

TransportationComplexFunction			
Code list proposed by the SIG 3D			
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	main through-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	dead-end 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		

TransportationComplexUsage
Code list identically specified as *TransportationComplexFunction*

C.7 Water body

WaterBodyClassType			
Code list proposed by the SIG 3D			
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

WaterBodyFunctionType			
Code list proposed by the SIG 3D			
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		

WaterBodyUsageType			
Codelist derived from Ruediger Drees und Andreas Kohlhaas			
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		

WaterLevelType			
Code list proposed by the SIG 3D			
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		

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 (base surface is restricted to a Polygon)
	TexturedSurface (defined in CityGML's *TexturedSurface* module, not in GML. This modelling approach has been marked deprecated. For restrictions see OrientableSurface)
	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 with 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**
core:CityModelType			0 – 4
	cityObjectMember	gml:FeaturePropertyType	0 – 4
	app:appearanceMember	app:AppearancePropertyType	0 – 4
	_GenericApplicationProperty OfCityModel	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
	app:appearance	app:AppearancePropertyType	0 – 4
	gen:_genericAttribute	gen:AbstractGenericAttributeType	0 – 4
	_GenericApplicationProperty OfCityObject	xs:anyType	0 – 4
core:AbstractSiteType			1 – 4
	_GenericApplicationProperty OfSite	xs:anyType	1 – 4
core:AddressType			0 – 4
	xalAddress	core:xalAddressPropertyType	0 – 4
	multiPoint	gml:MultiPointPropertyType	0 – 4
	_GenericApplicationProperty OfAddress	xs:anyType	0 – 4
app:AppearanceType			0 – 4
	theme	xs:string	0 – 4
	surfaceDataMember	app:SurfaceDataPropertyType	0 – 4
	_GenericApplicationProperty OfAppearance	xs:anyType	0 – 4
app:AbstractSurfaceDataType			0 – 4
	isFront	xs:Boolean	0 – 4
	_GenericApplicationProperty OfSurfaceData	xs:anyType	0 – 4
app:AbstractTextureType			0 – 4
	imageURI	xs:anyURI	0 – 4
	mimeType	core:MimeTypeType	0 – 4
	textureType	app:TextureTypeType	0 – 4
	wrapMode	app:WrapModeType	0 – 4
	borderColor	app:ColorPlusOpacity	0 – 4
	_GenericApplicationProperty OfTexture	xs:anyType	0 – 4
app:ParameterizedTextureType			0 – 4
	target	app:TextureAssociationType	0 – 4
	_GenericApplicationProperty OfParameterizedTexture	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
	_GenericApplicationProperty OfGeoreferencedTexture	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
	_GenericApplicationProperty OfX3DMaterial	xs:anyType	0 – 4
bldg:AbstractBuildingType			1 – 4
	class	bldg:BuildingClassType	1 – 4
	function	bldg:BuildingFunctionType	1 – 4
	usage	bldg:BuildingUsageType	1 – 4
	yearOfConstruction	xs:gYear	1 – 4
	yearOfDemolition	xs:gYear	1 – 4
	roofType	bldg:RoofTypeType	1 – 4
	measuredHeight	gml:LengthType	1 – 4
	storeysAboveGround	xs:nonNegativeInteger	1 – 4
	storeysBelowGround	xs:nonNegativeInteger	1 – 4
	storeyHeightsAboveGround	gml:MeasureOrNullListType	1 – 4
	storeyHeightsBelowGround	gml:MeasureOrNullListType	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
	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	1 – 4
	address	core:AddressPropertyType	1 – 4
	_GenericApplicationProperty OfAbstractBuilding	xs:anyType	1 – 4
bldg:BuildingType			1 – 4
	_GenericApplicationProperty OfBuiding	xs:anyType	1 – 4
bldg:BuildingPartType			1 – 4
	_GenericApplicationProperty OfBuidingPart	xs:anyType	1 – 4
bldg:BuildingInstallationType			2 – 4
	class	bldg:BuildingInstallationClassType	2 – 4
	function	bldg:BuildingInstallationFunctionType	2 – 4
	usage	bldg:BuildingInstallationUsageType	2 – 4
	lod2Geometry	gml:GeometryPropertyType	2
	lod3Geometry	gml:GeometryPropertyType	3
	lod4Geometry	gml:GeometryPropertyType	4
	_GenericApplicationProperty OfBuidingInstallation	xs:anyType	2 – 4

bldg:IntBuildingInstallationType			4
	class	bldg:IntBuildingInstallationClassType	4
	function	bldg:IntBuildingInstallationFunctionType	4
	usage	bldg:IntBuildingInstallationUsageType	4
	lod4Geometry	gml:GeometryPropertyType	4
	_GenericApplicationProperty OfIntBuidingInstallation	xs:anyType	4
bldg:AbstractBoundarySurface Type			2 – 4
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	opening	bldg:OpeningPropertyType	3 – 4
	_GenericApplicationProperty OfBoundarySurface	xs:anyType	2 – 4
bldg:RoofSurfaceType			2 – 4
	_GenericApplicationProperty OfRoofSurface	xs:anyType	2 – 4
bldg:WallSurfaceType			2 – 4
	_GenericApplicationProperty OfWallSurface	xs:anyType	2 – 4
bldg:GroundSurfaceType			2 – 4
	_GenericApplicationProperty OfGroundSurface	xs:anyType	2 – 4
bldg:ClosureSurfaceType			2 – 4
	_GenericApplicationProperty OfClosureSurface	xs:anyType	2 – 4
bldg:FloorSurfaceType			4
	_GenericApplicationProperty OfFloorSurface	xs:anyType	4
bldg:InteriorWallSurfaceType			4
	_GenericApplicationProperty OfInteriorWallSurface	xs:anyType	4
bldg:CeilingSurfaceType			4
	_GenericApplicationProperty OfCeilingSurface	xs:anyType	4
bldg:AbstractOpeningType			3 – 4
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	_GenericApplicationProperty OfOpening	xs:anyType	3 – 4
bldg:WindowType			3 – 4
	_GenericApplicationProperty OfWindow	xs:anyType	3 – 4
bldg:DoorType			3 – 4
	address	core:AddressPropertyType	3 – 4
	_GenericApplicationProperty OfCeilingSurface	xs:anyType	3 – 4
bldg:RoomType			4
	class	bldg:RoomClassType	4
	function	bldg:RoomFunctionType	4
	usage	bldg:RoomUsageType	4
	lod4Solid	gml:SolidPropertyType	4
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	boundedBy	bldg:BoundarySurfacePropertyType	4
	interiorFurniture	bldg:InteriorFurniturePropertyType	4
	roomInstallation	bldg:IntBuildingInstallationPropertyType	4
	_GenericApplicationProperty OfRoom	xs:anyType	4
bldg:BuildingFurnitureType			4
	class	bldg:BuildingFurnitureClassType	4
	function	bldg:BuildingFurnitureFunctionType	4
	usage	bldg:BuildingFurnitureUsageType	4
	lod4Geometry	gml:GeometryPropertyType	4
	lod4ImplicitRepresentation	bldg:ImplicitRepresentationPropertyType	4

	_GenericApplicationProperty OfBuildingFurniture	xs:anyType	4
frn:CityFurnitureType			1 – 4
	function	frn:FurnitureFunctionType	1 – 4
	class	frn:FurnitureClassType	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
	_GenericApplicationProperty OfCityFurniture	xs:anyType	1 – 4
grp:CityObjectGroupType			0 – 4
	function	xs:string	0 – 4
	class	xs:string	0 – 4
	usage	xs:string	0 – 4
	groupMember	grp:CityObjectGroupMemberType	0 – 4
	parent	grp:CityObjectGroupParentType	0 – 4
	geometry	gml:GeometryPropertyType	0 – 4
	_GenericApplicationProperty OfCityObjectGroup	xs:anyType	0 – 4
gen:GenericCityObjectType			0 – 4
	function	xs:string	0 – 4
	class	xs:string	0 – 4
	usage	xs:string	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
luse:LandUseType			0 – 4
	class	luse:LandUseClassType	0 – 4
	function	luse:LandUseFunctionType	0 – 4
	usage	luse:LandUseUsageType	0 – 4
	lod0MultiSurface	gml:MultiSurfacePropertyType	0
	lod1MultiSurface	gml:MultiSurfacePropertyType	1
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	_GenericApplicationProperty OfLandUse	xs:anyType	0 – 4
dem:ReliefFeatureType			0 – 4
	lod	core:integerBetween0and4	0 – 4
	reliefComponent	dem:ReliefComponentPropertyType	0 – 4
	_GenericApplicationProperty OfReliefFeature	xs:anyType	0 – 4

dem:AbstractReliefComponent Type			0 – 4
	lod	core:integerBetween0and4	0 – 4
	extent	gml:PolygonPropertyType	0 – 4
	_GenericApplicationProperty OfReliefComponent	xs:anyType	0 – 4
dem:TINReliefType			0 – 4
	tin	dem:tinPropertyType	0 – 4
	_GenericApplicationProperty OfTINRelief	xs:anyType	0 – 4
dem:RasterReliefType			0 – 4
	grid	dem:gridPropertyType	0 – 4
	_GenericApplicationProperty OfRasterRelief	xs:anyType	0 – 4
dem:MassPointReliefType			0 – 4
	reliefPoints	gml:MultiPointPropertyType	0 – 4
	_GenericApplicationProperty OfMassPointRelief	xs:anyType	0 – 4
dem:BreakLineReliefType			0 – 4
	ridgeOrValleyLines	gml:MultiCurvePropertyType	0 – 4
	breaklines	gml:MultiCurvePropertyType	0 – 4
	_GenericApplicationProperty OfBreakLineRelief	xs:anyType	0 – 4
tran:AbstractTransportationObject Type			0 – 4
	_GenericApplicationProperty OfTransportationObject	xs:anyType	0 – 4
tran:TransportationComplexType			0 – 4
	function	tran:TransportationComplexFunction Type	0 – 4
	usage	tran:TransportationComplexUsageType	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
	_GenericApplicationProperty OfTransportationComplex	xs:anyType	0 – 4
tran:TrafficAreaType			1 – 4
	usage	tran:TrafficAreaUsageType	1 – 4
	function	tran:TrafficAreaFunctionType	1 – 4
	surfaceMaterial	tran:TrafficSurfaceMaterialType	1 – 4
	lod1MultiSurface	gml:MultiSurfacePropertyType	1
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	_GenericApplicationProperty OfTrafficArea	xs:anyType	1 – 4
tran:AuxillaryTrafficAreaType			1 – 4
	function	tran:AuxiliaryTrafficAreaFunctionType	1 – 4
	surfaceMaterial	tran:TrafficSurfaceMaterialType	1 – 4
	lod1MultiSurface	gml:MultiSurfacePropertyType	1
	lod2MultiSurface	gml:MultiSurfacePropertyType	2
	lod3MultiSurface	gml:MultiSurfacePropertyType	3
	lod4MultiSurface	gml:MultiSurfacePropertyType	4
	_GenericApplicationProperty OfAuxiliaryTrafficArea	xs:anyType	1 – 4
tran:TrackType			1 – 4
	_GenericApplicationProperty OfTrack	xs:anyType	1 – 4
tran:RoadType			1 – 4
	_GenericApplicationProperty OfRoad	xs:anyType	1 – 4

tran:RailwayType			1 – 4
	_GenericApplicationProperty OfRailway	xs:anyType	1 – 4
tran:SquareType			1 – 4
	_GenericApplicationProperty OfSquare	xs:anyType	1 – 4
veg:AbstractVegetationObject Type			1 – 4
	_GenericApplicationProperty OfVegetationObject	xs:anyType	1 – 4
veg:PlantCoverType			1 – 4
	class	veg:PlantCoverClassType	1 – 4
	function	veg:PlantCoverFunctionType	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
	_GenericApplicationProperty OfPlantCover	xs:anyType	1 – 4
veg:SolitaryVegetation ObjectType			1 – 4
	class	veg:PlantClassType	1 – 4
	function	veg:PlantFunctionType	1 – 4
	species	veg:SpeciesType	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
	_GenericApplicationProperty OfSolitaryVegetationObject	xs:anyType	1 – 4
wtr:AbstractWaterObjectType			0 – 4
	_GenericApplicationProperty OfWaterObject	xs:anyType	0 – 4
wtr:WaterBodyType			0 – 4
	class	wtr:WaterBodyClassType	0 – 4
	function	wtr:WaterBodyFunctionType	0 – 4
	usage	wtr:WaterBodyUsageType	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:BoundedByWaterSurface PropertyType	2 – 4
	_GenericApplicationProperty OfWaterBody	xs:anyType	0 – 4
wtr:AbstractWaterBoundary SurfaceType			2 – 4
	lod2Surface	gml:SurfacePropertyType	2
	lod3Surface	gml:SurfacePropertyType	3
	lod4Surface	gml:SurfacePropertyType	4

	_GenericApplicationProperty OfWaterBoundarySurface	xs:anyType	2 – 4
wtr:WaterSurfaceType			2 – 4
	waterLevel	WaterLevelType	2 – 4
	_GenericApplicationProperty OfWaterSurface	xs:anyType	2 – 4
wtr:WaterGroundSurfaceType			2 – 4
	_GenericApplicationProperty OfWaterGroundSurface	xs:anyType	2 – 4
wtr:WaterClosureSurfaceType			2 – 4
	_GenericApplicationProperty OfWaterClosureSurface	xs:anyType	2 – 4

Annex F
(informative)

Example CityGML datasets

F.1 Example of a CityGML dataset for a building in LOD1 and LOD2

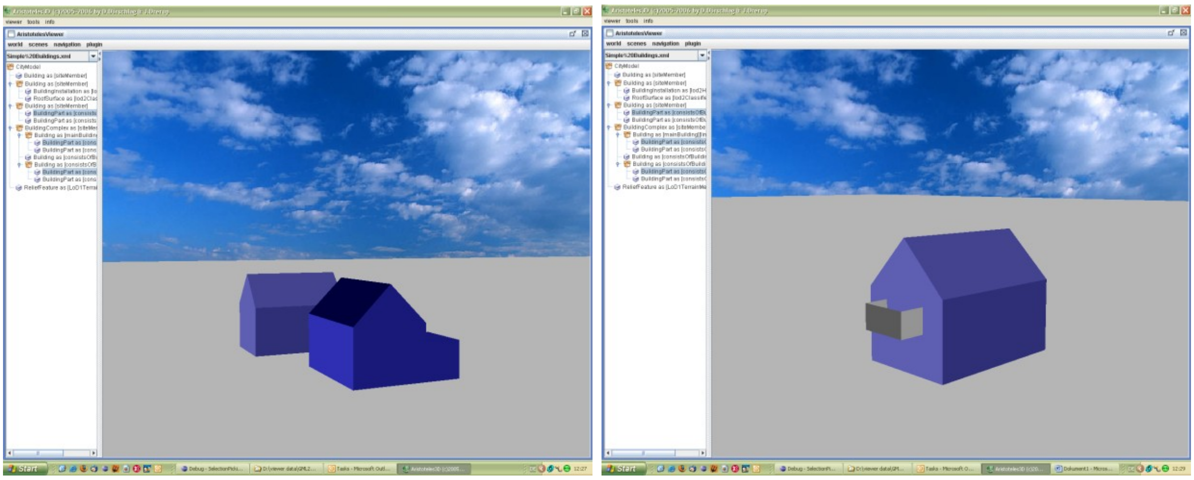


Fig. 50: Visualisation of the following CityGML dataset containing buildings in LOD1 and LOD2 (source: IGG Uni Bonn).


```xml
<?xml version="1.0" encoding="UTF-8"?>
<CityModel xmlns="http://www.opengis.net/citygml/1.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/1.0"
  xmlns:tex="http://www.opengis.net/citygml/texturedsurface/1.0"
  xmlns:grp="http://www.opengis.net/citygml/cityobjectgroup/1.0"
  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/building/1.0 http://www.citygml.org/citygml/building/1.0/building.xsd
  http://www.opengis.net/citygml/texturedsurface/1.0 http://www.citygml.org/citygml/texturedsurface/1.0/texturedSurface.xsd
  http://www.opengis.net/citygml/cityobjectgroup/1.0 http://www.citygml.org/citygml/cityobjectgroup/1.0/cityObjectGroup.xsd">
  <gml:description> Simple example for an XML dataset according to CityGML, the GML application schema of the SIG 3D. This
    dataset contains four parts with different complexities, which have been truncated here (the full version can be obtained
    from www.citygml.org):
    1.) Simple building in LOD2 with one textured and one colored surface
    2.) Simple building in LOD1 as blocks model without balcony, and the same building with gabled roof and balcony in LOD2.
    3.) House with gabled roof and garage, represented by two BuildingParts. The common wall surface of the building and the garage is
      defined only once and is in the boundary of one solid, and re-used by the second solid.
    4.) Building group consisting of two buildings that have been defined previously.
    The coordinate reference system is given in DHDN / Gauss-Krueger 3 degree (2nd zone) +
    normal heights above sea level (DHHN92). This system is referred to by
    srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783". Please note that the coordinates actually used in this
    dataset have been trimmed for clarity reasons and thus do not match this CRS. </gml:description>
  <gml:name>3D city model of Samplecity</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">0.0 0.0 0.0 </gml:pos>
      <gml:pos srsDimension="3">33.0 34.0 2.5</gml:pos>
    </gml:Envelope>
  </gml:boundedBy>
  <cityObjectMember>
    <!--Simple building with gabled roof with two storeys and an address. It is a LOD2 model, because it contains a roof shape.-->
    <bldg:Building gml:id="Build0815">
      <externalReference>
        <informationSystem>http://www.adv-online.de</informationSystem>
        <!-- Reference to the german cadastral database -->
        <externalObject>
          <uri>urn:adv:oid:DEHE123400007001</uri>
          <!-- ID of the object, being unique country-wide -->
        </externalObject>
```



```xml
        </externalReference>
        <bldg:function>1000</bldg:function>
        <bldg:yearOfConstruction>1985</bldg:yearOfConstruction>
        <bldg:roofType>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>
          <!--simple building with gabled roof-->
          <gml:Solid srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
            <gml:exterior>
              <gml:CompositeSurface>
                <gml:surfaceMember>
                  <tex:TexturedSurface orientation="+">
                    <!--front surface-->
                    <gml:baseSurface>
                      <gml:Polygon>
                        <gml:exterior>
                          <gml:LinearRing>
                            <gml:posList srsDimension="3"> 1.0 1.0 0.0 3.0 1.0 1.5 2.0 1.0 2.5 1.0 1.0 1.5
                              1.0 1.0 0.0 </gml:posList>
                          </gml:LinearRing>
                        </gml:exterior>
                      </gml:Polygon>
                    </gml:baseSurface>
                    <tex:appearance>
                      <tex:SimpleTexture>
                        <tex:textureMap>FrontTexture096454.jpg</tex:textureMap>
                        <tex:textureCoordinates> 0.05 0.07 0.95 0.07 0.95 0.5 0.5 1 0.05 0.5 0.05 0.07 </tex:textureCoordinates>
                        <tex:textureType>specific</tex:textureType>
                      </tex:SimpleTexture>
                    </tex:appearance>
                  </tex:TexturedSurface>
                </gml:surfaceMember>
                <gml:surfaceMember>
                  <tex:TexturedSurface orientation="+">
                    <!--back surface-->
                    <gml:baseSurface>
                      <gml:Polygon>
                        <gml:exterior>
                          <gml:LinearRing>
                            <gml:pos srsDimension="3">1.0 4.0 0.0</gml:pos>
                            <gml:pos srsDimension="3">1.0 4.0 1.5</gml:pos>
                            <gml:pos srsDimension="3">2.0 4.0 2.5</gml:pos>
                            <gml:pos srsDimension="3">3.0 4.0 1.5</gml:pos>
                            <gml:pos srsDimension="3">3.0 4.0 0.0</gml:pos>
                            <gml:pos srsDimension="3">1.0 4.0 0.0</gml:pos>
                          </gml:LinearRing>
                        </gml:exterior>
                      </gml:Polygon>
                    </gml:baseSurface>
                    <tex:appearance>
                      <tex:Material>
                        <tex:ambientIntensity>0.4</tex:ambientIntensity>
                        <tex:diffuseColor> 0 0 1 </tex:diffuseColor>
                        <!-- defines blue color -->
                      </tex:Material>
                    </tex:appearance>
                  </tex:TexturedSurface>
                </gml:surfaceMember>
                …
              </gml:CompositeSurface>
            </gml:exterior>
          </gml:Solid>
        </bldg:lod2Solid>
      </bldg:Building>
    </cityObjectMember>
    <cityObjectMember>
      <!--  Simple building represented in LOD1 (as blocks model without balcony) and in LOD2
        with roof shape and balcony. One of the roof surfaces is represented explicitly as a thematic surface object (RoofSurface).
        The function is residential building (1000) and the roof type is 'gabled roof' (1030).
        Both values are defined in external code lists.-->
      <bldg:Building gml:id="Build0816">
        <gml:name>Villa Kunterbunt</gml:name>
        <bldg:function>1000</bldg:function>
        <bldg:yearOfConstruction>1952</bldg:yearOfConstruction>
```



```xml
<bldg:roofType>1030</bldg:roofType>
<bldg:lod1Solid>
  <gml:Solid srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
    <!--simple blocks model -->
    <gml:exterior>
      <gml:CompositeSurface>
        <gml:surfaceMember>
          <!--front surface-->
          <gml:Polygon>
            <gml:exterior>
              <gml:LinearRing>
                <gml:pos srsDimension="3">31.0 31.0 0.0</gml:pos>
                <gml:pos srsDimension="3">33.0 31.0 0.0</gml:pos>
                <gml:pos srsDimension="3">33.0 31.0 1.5</gml:pos>
                <gml:pos srsDimension="3">31.0 31.0 1.5</gml:pos>
                <gml:pos srsDimension="3">31.0 31.0 0.0</gml:pos>
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </gml:surfaceMember>
        …
      </gml:CompositeSurface>
    </gml:exterior>
  </gml:Solid>
</bldg:lod1Solid>
<bldg:lod2Solid>
  <gml:Solid srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
    <!-- simple building with gabled roof-->
    <gml:exterior>
      <gml:CompositeSurface>
        <gml:surfaceMember>
          <!--front surface-->
          <gml:Polygon>
            <gml:exterior>
              <gml:LinearRing>
                <gml:pos srsDimension="3">31.0 31.0 0.0</gml:pos>
                <gml:pos srsDimension="3">33.0 31.0 0.0</gml:pos>
                <gml:pos srsDimension="3">33.0 31.0 1.5</gml:pos>
                <gml:pos srsDimension="3">32.0 31.0 2.5</gml:pos>
                <gml:pos srsDimension="3">31.0 31.0 1.5</gml:pos>
                <gml:pos srsDimension="3">31.0 31.0 0.0</gml:pos>
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </gml:surfaceMember>
        <gml:surfaceMember>
          <!--1st roof surface. This polygon will be referenced below.-->
          <gml:Polygon gml:id="roofsurface4711">
            <gml:exterior>
              <gml:LinearRing>
                <gml:posList srsDimension="3"> 32.0 31.0 2.5 33.0 31.0 1.5 33.0 34.0 1.5 32.0 34.0 2.5
                  32.0 31.0 2.5 </gml:posList>
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </gml:surfaceMember>
        …
      </gml:CompositeSurface>
    </gml:exterior>
  </gml:Solid>
</bldg:lod2Solid>
<bldg:outerBuildingInstallation>
  <bldg:BuildingInstallation>
    <gml:name>The nice balcony to the south</gml:name>
    <bldg:function>1000</bldg:function>
    <!--function 1000 of a BuildingInstallation means 'balcony'-->
    <bldg:lod2Geometry>
      <!-- The balcony is situated at the 1st  front surface -->
      <!-- The geometry of the balcony is defined by an aggregation of 3D surfaces. -->
      <gml:CompositeSurface srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
        <gml:surfaceMember>
          <!-- ground surface of the balcony -->
          <gml:Polygon>
            <gml:exterior>
              <gml:LinearRing>
                <gml:pos srsDimension="3">31.5 30.5 0.8</gml:pos>
```



```xml
                        <gml:pos srsDimension="3">31.5 31.0 0.8</gml:pos>
                        <gml:pos srsDimension="3">32.5 31.0 0.8</gml:pos>
                        <gml:pos srsDimension="3">32.5 30.5 0.8</gml:pos>
                        <gml:pos srsDimension="3">31.5 30.5 0.8</gml:pos>
                      </gml:LinearRing>
                    </gml:exterior>
                  </gml:Polygon>
                </gml:surfaceMember>
                …
              </gml:CompositeSurface>
            </bldg:lod2Geometry>
          </bldg:BuildingInstallation>
        </bldg:outerBuildingInstallation>
        <bldg:boundedBy>
          <bldg:RoofSurface>
            <externalReference>
              <informationSystem>http://www.solar-panel.com/database/samplecity</informationSystem>
              <!--  This may be a database, which contains all roof surfaces of a city covered with solar panels -->
              <externalObject>
                <name>roof_10786</name>
                <!-- roof_10786 is the id of the roof surface in the external solar panel database  -->
              </externalObject>
            </externalReference>
            <bldg:lod2MultiSurface>
              <gml:MultiSurface>
                <!-- Reference to a surface which has already been defined in the solid boundary of the outer building shell.-->
                <gml:surfaceMember xlink:href="#roofsurface4711"/>
              </gml:MultiSurface>
            </bldg:lod2MultiSurface>
          </bldg:RoofSurface>
        </bldg:boundedBy>
      </bldg:Building>
    </cityObjectMember>
    <cityObjectMember>
      <!-- House with gabled roof and a garage, represented by two BuildingParts. The common wall surface of the building and the
       garage is shared by both solids realizing a topological connection between both parts.-->
      <bldg:Building gml:id="Build0817">
        <bldg:consistsOfBuildingPart>
          <bldg:BuildingPart gml:id="Build0817a">
            <bldg:function>1000</bldg:function>
            <bldg:yearOfConstruction>1964</bldg:yearOfConstruction>
            <bldg:roofType>1030</bldg:roofType>
            <bldg:storeysAboveGround>2</bldg:storeysAboveGround>
            <bldg:lod2Solid>
              <gml:Solid srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
                <!--Building with gabled roof-->
                <gml:exterior>
                  <gml:CompositeSurface>
                    <gml:surfaceMember>
                      <!--front surface-->
                      <gml:Polygon>
                        <gml:exterior>
                          <gml:LinearRing>
                            <gml:posList srsDimension="3"> 8.0 2.0 0.0 8.0 4.0 0.0 8.0 4.0 1.5 8.0 3.0 2.5
                              8.0 2.0 1.5 8.0 2.0 0.0 </gml:posList>
                          </gml:LinearRing>
                        </gml:exterior>
                      </gml:Polygon>
                    </gml:surfaceMember>
                    …
                    <gml:surfaceMember>
                      <!--2nd side surface, shares surface with garage geometry-->
                      <gml:Polygon gml:id="polygon007">
                        <gml:exterior>
                          <gml:LinearRing>
                            <gml:pos srsDimension="3">8.0 4.0 1.0</gml:pos>
                            <gml:pos srsDimension="3">8.0 4.0 0.0</gml:pos>
                            <gml:pos srsDimension="3">6.5 4.0 0.0</gml:pos>
                            <gml:pos srsDimension="3">6.5 4.0 1.0</gml:pos>
                            <gml:pos srsDimension="3">8.0 4.0 1.0</gml:pos>
                          </gml:LinearRing>
                        </gml:exterior>
                      </gml:Polygon>
                    </gml:surfaceMember>
                  </gml:CompositeSurface>
                  …
```



```xml
          </gml:exterior>
        </gml:Solid>
      </bldg:lod2Solid>
    </bldg:BuildingPart>
  </bldg:consistsOfBuildingPart>
  <bldg:consistsOfBuildingPart>
    <bldg:BuildingPart gml:id="Build817b">
      <bldg:function>1630</bldg:function>
      <!-- Function 1630 means 'garage' -->
      <bldg:yearOfConstruction>1996</bldg:yearOfConstruction>
      <bldg:roofType>1000</bldg:roofType>
      <bldg:storeysAboveGround>1</bldg:storeysAboveGround>
      <bldg:lod2Solid>
        <gml:Solid srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
          <!--garage-->
          <gml:exterior>
            <gml:CompositeSurface>
              <gml:surfaceMember>
                <!--front surface-->
                <gml:Polygon>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:pos srsDimension="3">8.0 4.0 0.0</gml:pos>
                      <gml:pos srsDimension="3">8.0 5.0 0.0</gml:pos>
                      <gml:pos srsDimension="3">8.0 5.0 1.0</gml:pos>
                      <gml:pos srsDimension="3">8.0 4.0 1.0</gml:pos>
                      <gml:pos srsDimension="3">8.0 4.0 0.0</gml:pos>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
              …
              <gml:surfaceMember>
                <!--2nd side surface, shares surface with building geometry-->
                <gml:OrientableSurface orientation="-">
                  <!-- Surface orientation has to be reversed! -->
                  <gml:baseSurface xlink:href="#polygon007"/>
                </gml:OrientableSurface>
              </gml:surfaceMember>
            </gml:CompositeSurface>
          </gml:exterior>
        </gml:Solid>
      </bldg:lod2Solid>
    </bldg:BuildingPart>
  </bldg:consistsOfBuildingPart>
  <bldg:address>
    <Address>
      <xalAddress>
        <xAL:AddressDetails>
          <xAL:Country>
            <xAL:CountryName>Germany</xAL:CountryName>
            <xAL:Locality Type="Town">
              <xAL:LocalityName>Bonn</xAL:LocalityName>
              <xAL:Thoroughfare Type="Street">
                <xAL:ThoroughfareNumber>172</xAL:ThoroughfareNumber>
                <xAL:ThoroughfareName>Meckenheimer Allee</xAL:ThoroughfareName>
              </xAL:Thoroughfare>
              <xAL:PostalCode>
                <xAL:PostalCodeNumber>53115</xAL:PostalCodeNumber>
              </xAL:PostalCode>
            </xAL:Locality>
          </xAL:Country>
        </xAL:AddressDetails>
      </xalAddress>
      <multiPoint>
        <gml:MultiPoint>
          <gml:pointMember>
            <gml:Point>
              <gml:pos srsDimension="3">6.5 4.0 1.0</gml:pos>
            </gml:Point>
          </gml:pointMember>
        </gml:MultiPoint>
      </multiPoint>
    </Address>
  </bldg:address>
</bldg:Building>
```

```xml
  </cityObjectMember>
  <cityObjectMember>
    <!--Building group with name 'Scenic view', consisting of the two buildings Build0815 and Build0817.
     Both buildings are included by reference.-->
    <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>
  </cityObjectMember>
</CityModel>
```

Listing 1:Excerpt from the CityGML dataset for buildings in LOD1 and 2 visualised in Fig. 50.

F.2 Example of a CityGML dataset for a building in LOD3

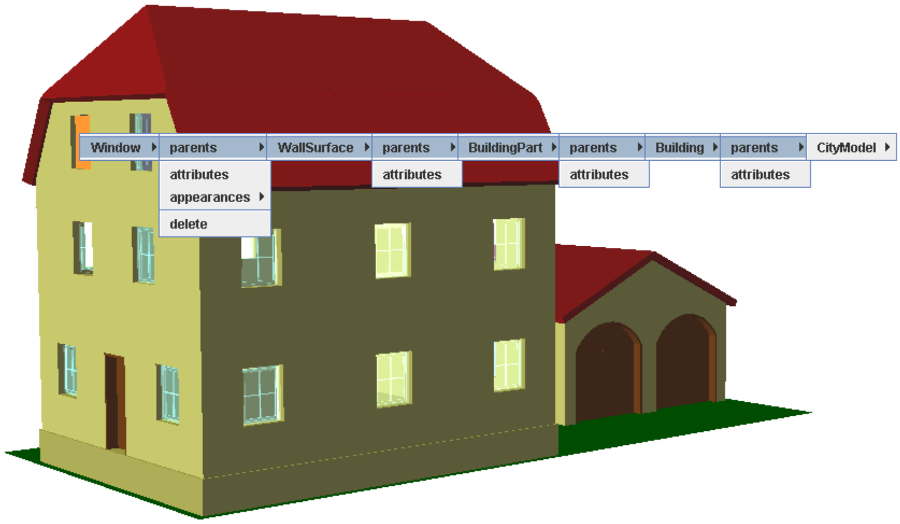


Fig. 51: Visualisation of buildings in LOD3, automatically generated from IFC building objects. Please note the coherent semantic and geometric decomposition (source: Research Center Karlsruhe).

```
<?xml version="1.0" encoding="UTF-8"?>
<CityModel xmlns="http://www.opengis.net/citygml/1.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/1.0"
  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/building/1.0 http://www.citygml.org/citygml/building/1.0/building.xsd">
  <gml:description>This file contains four buildings which are automatically converted from IFC models. This listing only shows
    an excerpt. The full dataset can be downloaded from http://www.citygml.org
    (example dataset for “four buildings in LOD3”)</gml:description>
  <gml:name>IFC_Building_Variant</gml:name>
  <gml:boundedBy>
    <gml:Envelope srsName="urn:ogc:def:crs,crs:EPSG:6.12:31467,crs:EPSG:6.12:5783">
      <gml:pos srsDimension="3">5429999.751795 3449999.751795 0.0</gml:pos>
      <gml:pos srsDimension="3">5430023.2 3450021.2 20.0</gml:pos>
    </gml:Envelope>
  </gml:boundedBy>
  …
  <cityObjectMember>
    <bldg:Building gml:id="GEB_TH_IFC_Building_Variant_GEB_75">
      <gml:description>Building in LOD3</gml:description>
      <gml:name>Building-ADT-2006</gml:name>
      <externalReference>
        <informationSystem>http://www.iai.fzk.de/raw/pages/german/projekte/VR-Systeme/html/Download/</informationSystem>
        <externalObject>
          <uri>urn:ifc:oid:0deJpNQ05BvwV03c405oVp</uri>
        </externalObject>
      </externalReference>
      <bldg:boundedBy>
        <bldg:RoofSurface gml:id="GEB_TH_IFC_Building_Variant_DACH_136">
          <externalReference>
            <informationSystem>http://www.iai.fzk.de/raw/pages/german/projekte/VR-Systeme/html/Download/</informationSystem>
            <externalObject>
              <uri>urn:ifc:oid:3CPSkwS7f9QRfhfr5gf7dq</uri>
            </externalObject>
          </externalReference>
          <bldg:lod3MultiSurface>
            <gml:MultiSurface>
              <gml:surfaceMember>
                <gml:Polygon>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:posList srsDimension="3">5430006.994499969 3449999.850802998 9.141580054626465
                        5430007.093499946 3449999.7517950004 8.970100114212036 5430000.906494903
                        3449999.7517950004 8.970100114212036 5430001.005499649 3449999.850802998
                        9.141580054626465 5430003.999999809 3450000.9735459564 11.086200187072754
                        5430006.994499969 3449999.850802998 9.141580054626465</gml:posList>
```



```xml
                                </gml:LinearRing>
                            </gml:exterior>
                        </gml:Polygon>
                    </gml:surfaceMember>
                    <gml:surfaceMember>
                        <gml:Polygon>
                            <gml:exterior>
                                <gml:LinearRing>
                                    <gml:posList srsDimension="3">5430006.920299816 3449999.925 8.870099971160888
                                        5430006.845300007 3450000.000000003 8.999999949798584 5430003.999999809
                                        3450001.066800046 10.847800204620361 5430001.154700088 3450000.000000003
                                        8.999999949798584 5430001.079700279 3449999.925 8.870099971160888
                                        5430006.920299816 3449999.925 8.870099971160888</gml:posList>
                                </gml:LinearRing>
                            </gml:exterior>
                        </gml:Polygon>
                    </gml:surfaceMember>
                    …
                </gml:MultiSurface>
            </bldg:lod3MultiSurface>
        </bldg:RoofSurface>
    </bldg:boundedBy>
    <bldg:boundedBy>
        <bldg:WallSurface gml:id="GEB_TH_IFC_Building_Variant_WAND_78">
            <externalReference>
                <informationSystem>http://www.iai.fzk.de/raw/pages/german/projekte/VR-Systeme/html/Download/</informationSystem>
                <externalObject>
                    <uri>urn:ifc:oid:2es$8LnAD9UxRIGzY8UaVK</uri>
                </externalObject>
            </externalReference>
            <bldg:lod3MultiSurface>
                <gml:MultiSurface>
                    <gml:surfaceMember>
                        <gml:Polygon>
                            <gml:exterior>
                                <gml:LinearRing>
                                    <gml:posList srsDimension="3">5429999.999999809 3450004.4950001715 6.059999996886253
                                        5429999.999999809 3450004.4950001715 4.800000021324157 5430000.119999695
                                        3450004.4950001715 4.800000021324157 5430000.180000114 3450004.4950001715
                                        4.800000021324157 5430000.3 3450004.4950001715 4.800000021324157 5430000.3
                                        3450004.4950001715 6.059999996886253 5430000.180000114 3450004.4950001715
                                        6.059999996886253 5430000.119999695 3450004.4950001715 6.059999996886253
                                        5429999.999999809 3450004.4950001715 6.059999996886253</gml:posList>
                                </gml:LinearRing>
                            </gml:exterior>
                        </gml:Polygon>
                    </gml:surfaceMember>
                    …
                </gml:MultiSurface>
            </bldg:lod3MultiSurface>
            <bldg:opening>
                <bldg:Window gml:id="GEB_TH_IFC_Building_Variant_OEFF_OBJ_80">
                    <externalReference>
                        <informationSystem>
                            http://www.iai.fzk.de/raw/pages/german/projekte/VR-Systeme/html/Download/
                        </informationSystem>
                        <externalObject>
                            <uri>urn:ifc:oid:3VkZRUoa97GgMdD342zHck</uri>
                        </externalObject>
                    </externalReference>
                    <bldg:lod3MultiSurface>
                        <gml:MultiSurface>
                            <gml:surfaceMember>
                                <gml:Polygon>
                                    <gml:exterior>
                                        <gml:LinearRing>
                                            <gml:posList srsDimension="3">5430000.119999695 3450008.940000343
                                                2.9999999497985836 5430000.180000114 3450008.940000343 2.9999999497985836
                                                5430000.180000114 3450008.940000343 1.920000026092529 5430000.180000114
                                                3450008.940000343 1.860000083312988 5430000.119999695 3450008.940000343
                                                1.860000083312988 5430000.119999695 3450008.940000343
                                            2.9999999497985836</gml:posList>
                                        </gml:LinearRing>
                                    </gml:exterior>
                                </gml:Polygon>
                            </gml:surfaceMember>
```



```
                    ...
                    </gml:MultiSurface>
                </bldg:lod3MultiSurface>
            </bldg:Window>
        </bldg:opening>
        ...
    </bldg:WallSurface>
  </bldg:boundedBy>
 </bldg:Building>
</cityObjectMember>
</CityModel>
```

Listing 2:Excerpt from the CityGML dataset for the buildings in LOD3 visualised in Fig. 51.

F.3 Example of a CityGML dataset illustrating the appearance model

The following CityGML dataset contains a simple building given in geometric representations for 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 modelling approach results in four possible visualizations of the dataset that are represented in Fig. 52 and Fig. 53.

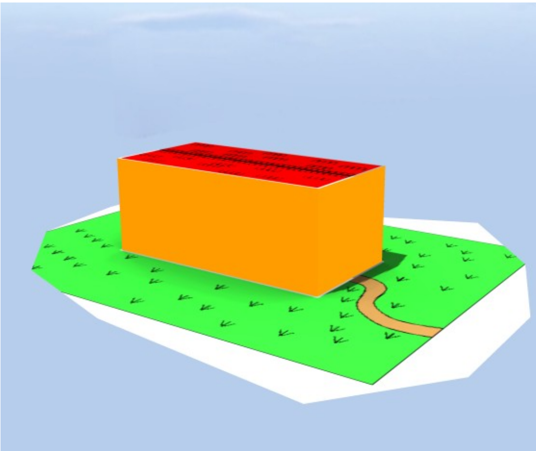
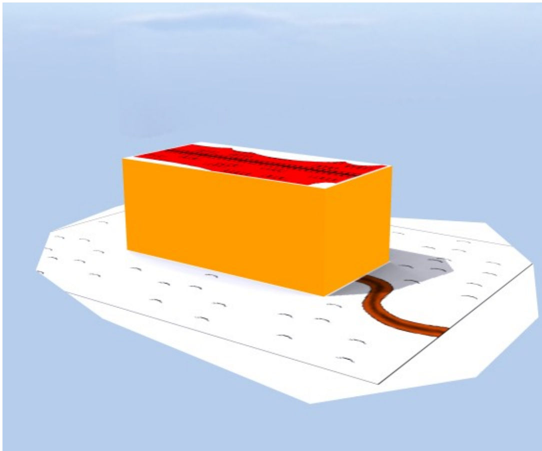
a.  b. 

Fig. 52: 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 time and (b) showing the building in winter time (image: Hasso-Plattner-Institute).

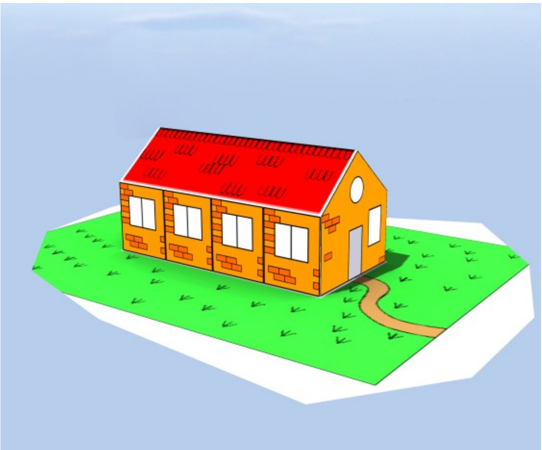
a.  b. 

Fig. 53: 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 time and (b) showing the building in winter time (image: Hasso-Plattner-Institute).

```
<?xml version="1.0" encoding="UTF-8"?>
<CityModel xmlns="http://www.opengis.net/citygml/1.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/1.0"
  xmlns:app="http://www.opengis.net/citygml/appearance/1.0"
  xmlns:dem="http://www.opengis.net/citygml/relief/1.0"
  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/building/1.0 http://www.citygml.org/citygml/building/1.0/building.xsd
  http://www.opengis.net/citygml/appearance/1.0 http://www.citygml.org/citygml/appearance/1.0/appearance.xsd
```



```xml
http://www.opengis.net/citygml/relief/1.0 http://www.citygml.org/citygml/relief/1.0/relief.xsd">
<gml:description> Simple example for an XML dataset according to CityGML, the GML application schema of the SIG 3D. This
  dataset contains one simple building in LOD1 and LOD2 and the surrounding terrain as well as two separate appearance
  themes:
  1.) Simple building in LOD1
  2.) Simple building in LOD2
  3.) Digital terrain given by a TIN.
  4.) Appearance theme “summer”.
  5.) Appearance theme “winter”.
  Please note, that appearances are explicitly linked to GML geometry objects using
  URIs. Since Texture objects are modelled as features (with a unique id) they can be (and in fact are) reused. This is
  realized using XLinks. The coordinate reference system is given in DHDN / Gauss-Krueger 3 degree (2nd zone) + normal
  heights above sea level (DHHN92). This system is referred to by
  srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783". Please note, that the coordinates actually used in this
  dataset have been trimmed for clarity reasons and thus do not match this CRS. </gml:description>
<gml:boundedBy>
  <gml:Envelope srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783">
    <gml:lowerCorner>-6.0 -7.0 0.0</gml:lowerCorner>
    <gml:upperCorner>17.0 13.0 5.0</gml:upperCorner>
  </gml:Envelope>
</gml:boundedBy>
<cityObjectMember>
  <bldg:Building gml:id="Build0815">
    <bldg:yearOfConstruction>2007</bldg:yearOfConstruction>
    <bldg:measuredHeight uom="#m">5.0</bldg:measuredHeight>
    <bldg:lod1Solid>
      <gml:Solid>
        <gml:exterior>
          <gml:CompositeSurface gml:id="lod1Surface">
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList srsDimension="3"> 0.0 0.0 0.0 10.0 0.0 0.0 10.0 0.0 4.0 0.0 0.0 4.0 0.0
                      0.0 0.0 </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"> 0.0 0.0 4.0 10.0 0.0 4.0 10.0 5.0 4.0 0.0 5.0 4.0 0.0
                      0.0 4.0 </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"> 0.0 0.0 0.0 5.0 0.0 0.0 5.0 0.0 3.0 0.0 0.0 3.0
                          0.0 0.0 0.0 </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"> 5.0 0.0 0.0 10.0 0.0 0.0 10.0 0.0 3.0 5.0 0.0
                          3.0 5.0 0.0 0.0 </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"> 10.0 0.0 0.0 10.0 5.0 0.0 10.0 5.0 3.0 10.0 2.5 5.0
                    10.0 0.0 3.0 10.0 0.0 0.0 </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"> 10.0 5.0 0.0 0.0 5.0 0.0 0.0 5.0 3.0 10.0 5.0 3.0 10.0
                    5.0 0.0 </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"> 0.0 5.0 0.0 0.0 0.0 0.0 0.0 0.0 3.0 0.0 2.5 5.0 0.0 5.0
                    3.0 0.0 5.0 0.0 </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"> 0.0 0.0 3.0 10.0 0.0 3.0 10.0 2.5 5.0 0.0 2.5 5.0 0.0
                    0.0 3.0 </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"> 10.0 5.0 3.0 0.0 5.0 3.0 0.0 2.5 5.0 10.0 2.5 5.0 10.0
                    5.0 3.0 </gml:posList>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
        </gml:CompositeSurface>
      </gml:exterior>
    </gml:Solid>
  </bldg:lod2Solid>
 </bldg:Building>
</cityObjectMember>
<cityObjectMember>
  <dem:ReliefFeature gml:id="DTM_1">
    <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> -6.0 5.0 0.0 0.0 5.0 0.0 -6.0 9.0 0.0 -6.0 5.0 0.0 </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> -5.0 -5.0 </gml:pos>
            </gml:Point>
          </app:referencePoint>
          <app:orientation> 0.05 0.0 0.0 0.066667 </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 1.0 0.0 0.0 0.3333 0.0 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> -5.0 -5.0 </gml:pos>
            </gml:Point>
          </app:referencePoint>
          <app:orientation> 0.05 0.0 0.0 0.066667 </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 3: Excerpt from the CityGML dataset illustrating CityGML's appearance model. The dataset is visualised in Fig. 52 and Fig. 53.

The following three raster images (Fig. 54 - Fig. 56) 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. 54) is assigned to the side surfaces using the texture wrapping mode *wrap* and is applied both within the summer and the winter theme.

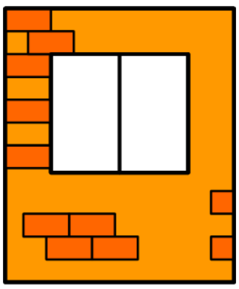


Fig. 54: Image *facade.png* used in the dataset to texturize the side surfaces of the building in LOD2 (cf. Fig. 53 a. and b.) (image: Hasso-Plattner-Institute).

Fig. 55 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.

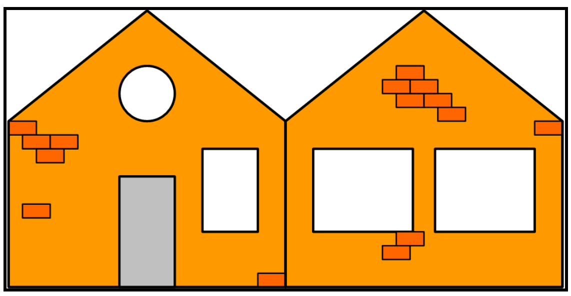


Fig. 55: Texture atlas *front_back_summer.png* containing the textures for the front surface and the back surface of the building in LOD2 within the summer theme (cf. Fig. 53 a.) (image: Hasso-Plattner-Institute).

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.

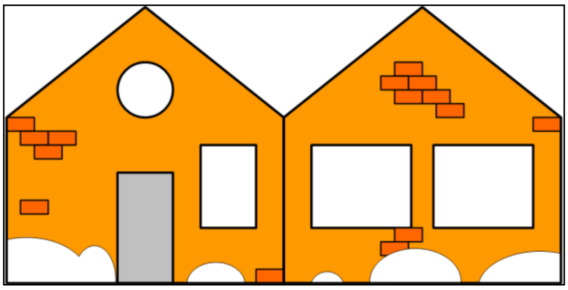


Fig. 56: Texture atlas *front_back_winter.png* containing the textures for the front surface and the back surface of the building in LOD2 within the winter theme (cf. Fig. 53 b.) (image: Hasso-Plattner-Institute).

The raster images shown in Fig. 57 and Fig. 58 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. 57) represents the texture for the summer theme, *ground_winter.png* (cf. Fig. 58) is used within the winter theme.

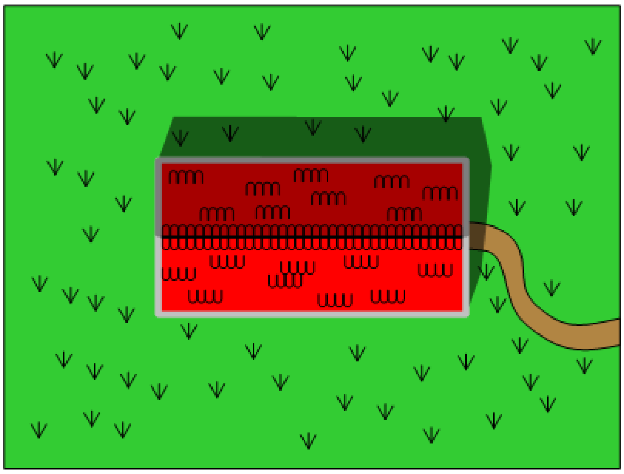


Fig. 57: The image *ground_summer.png* is assigned to the terrain and the roof surfaces of the building both in LOD1 and LOD2 (cf. Fig. 52 a. and Fig. 53 a.) within the summer theme (image: Hasso-Plattner-Institute).

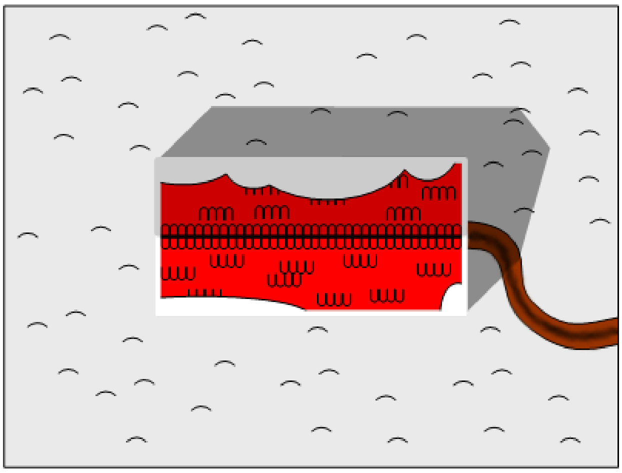


Fig. 58: The image *ground_winter.png* is assigned to the terrain and the roof surfaces of the building both in LOD1 and LOD2 (cf. Fig. 52 b. and Fig. 53 b.) within the winter theme (image: Hasso-Plattner-Institute).

F.4 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. 59). It uses two *app:ParameterizedTexture* objects, a regular road piece (`rd`, Fig. 60) and a dirt track becoming a paved road (`dt`, Fig. 61). 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. 62) or a texture for the complete roundabout (Fig. 63). In both cases, texture space is wasted. Wasted areas are marked red.

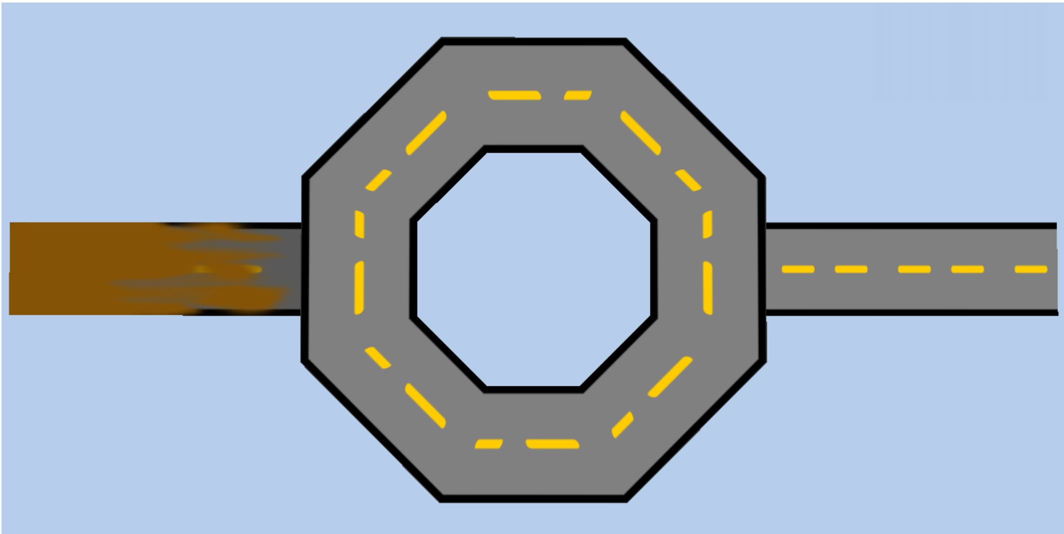


Fig. 59: A rendering of the textured geometry (images for this example: Hasso-Plattner-Institute).

Fig. 60: The road piece texture.

Fig. 61: The dirt track texture.

Fig. 62: A distortion-free road piece for the roundabout.
Red areas never become visible and are wasted.

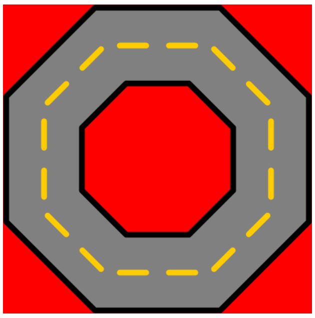


Fig. 63: The complete roundabout in a distortion-free texture.
Red areas never become visible and are wasted.

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/1.0"
  xmlns:app="http://www.opengis.net/citygml/appearance/1.0"
  xmlns:tran="http://www.opengis.net/citygml/transportation/1.0"
  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/1.0 http://www.citygml.org/citygml/appearance/1.0/appearance.xsd
  http://www.opengis.net/citygml/transportation/1.0 http://www.citygml.org/citygml/transportation/1.0/transportation.xsd">
  <gml:boundedBy>
    <gml:Envelope>
      <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>
```



```xml
                <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 4: CityGML dataset illustrating the use of texture coordinates for complex surfaces with holes. The dataset is visualised in Fig. 59.

Annex G
(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. 64). These noise maps are generated on the basis of acoustic models and noise propagation calculations, not on the basis of measurements (Fig. 65). 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).

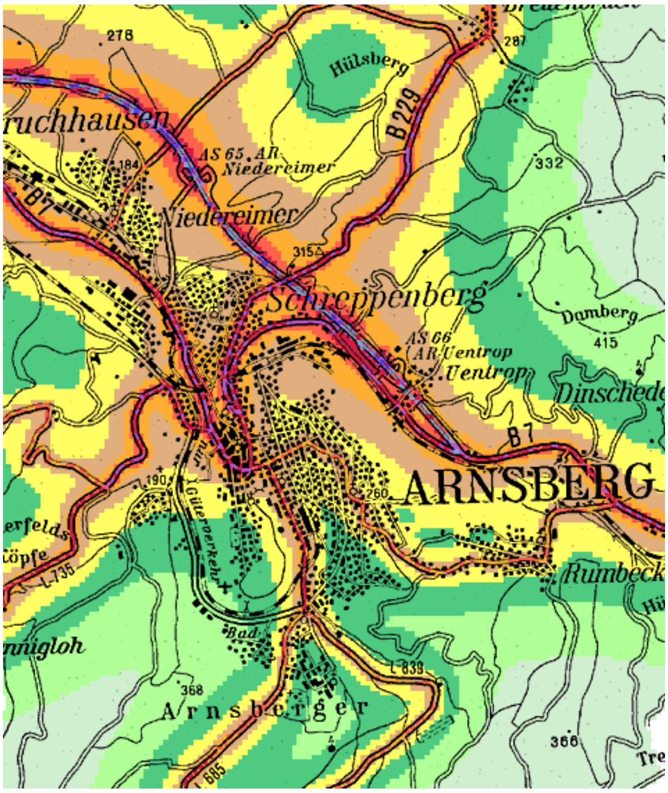


Fig. 64: Noise map generated from the 3D CityGML geodata and used for the reporting demanded of the EU Environmental Noise Directive (dark colours show higher noise immission) (source: State Agency for nature, environment and consumer protection NRW).

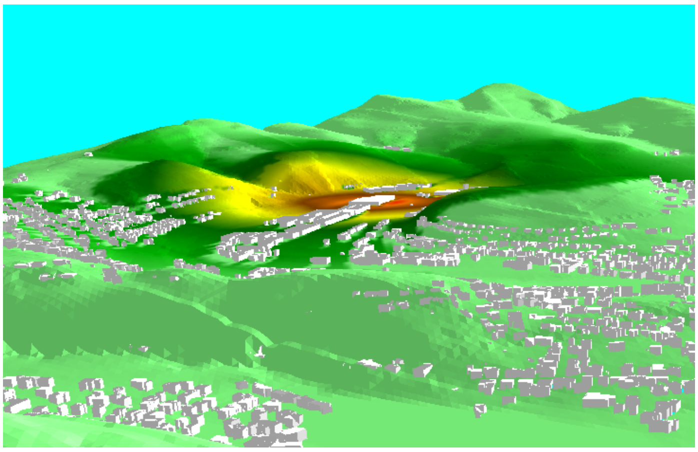


Fig. 65: Modelling a noise emission source using the 3D CityGML geodata in a special noise calculation software as first step before generating the noise map in Fig. 64 (source: Surveying and Mapping Agency NRW, Stapelfeldt GmbH, Institute of Geodesy and Geoinformation Uni Bonn).

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).

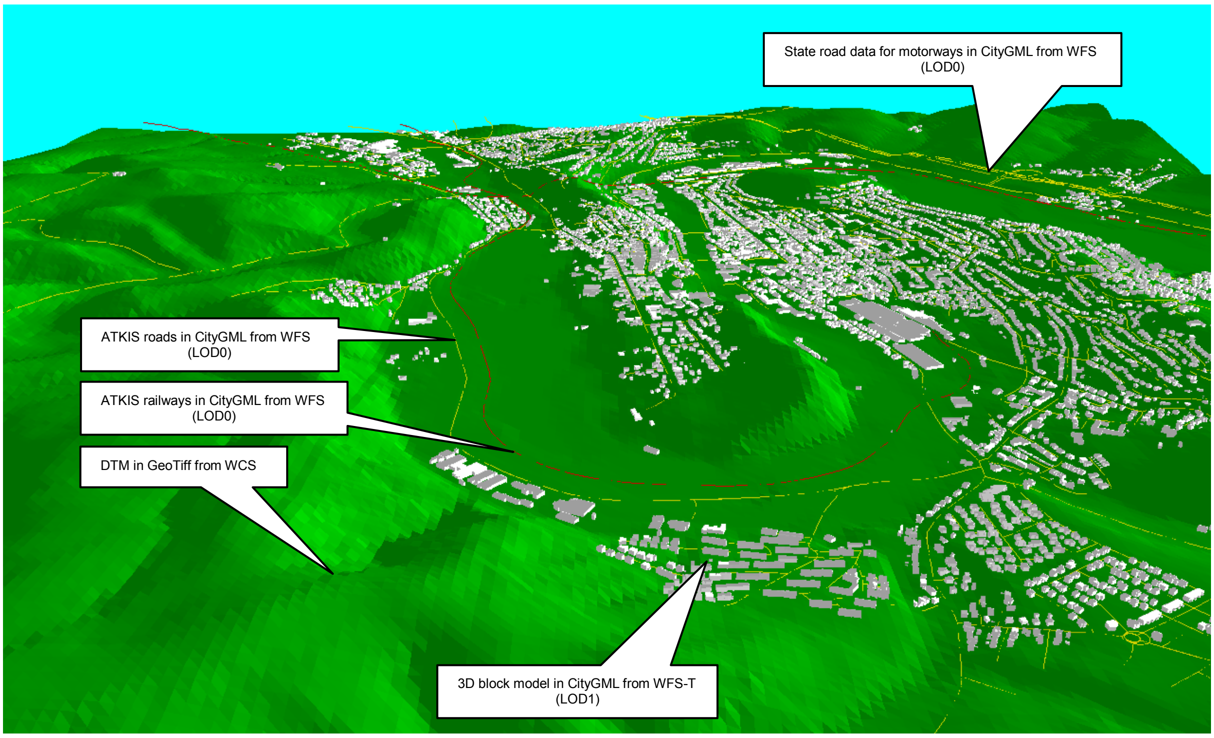


Fig. 66: 3D geodata in CityGML for the calculation of the noise map in Fig. 64: DTM in GeoTiff, 3D block model in CityGML, 3D road and railway data in CityGML, state road data for higher-level roads in CityGML (source: Surveying and Mapping Agency NRW, State Road Enterprice NRW, Stapelfeldt GmbH, Institute 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. 66 -Fig. 68). 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 10.11). 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. 69) as well as noise attributes attached to existing objects (e.g. reflection of buildings, Fig. 70). 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. 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).

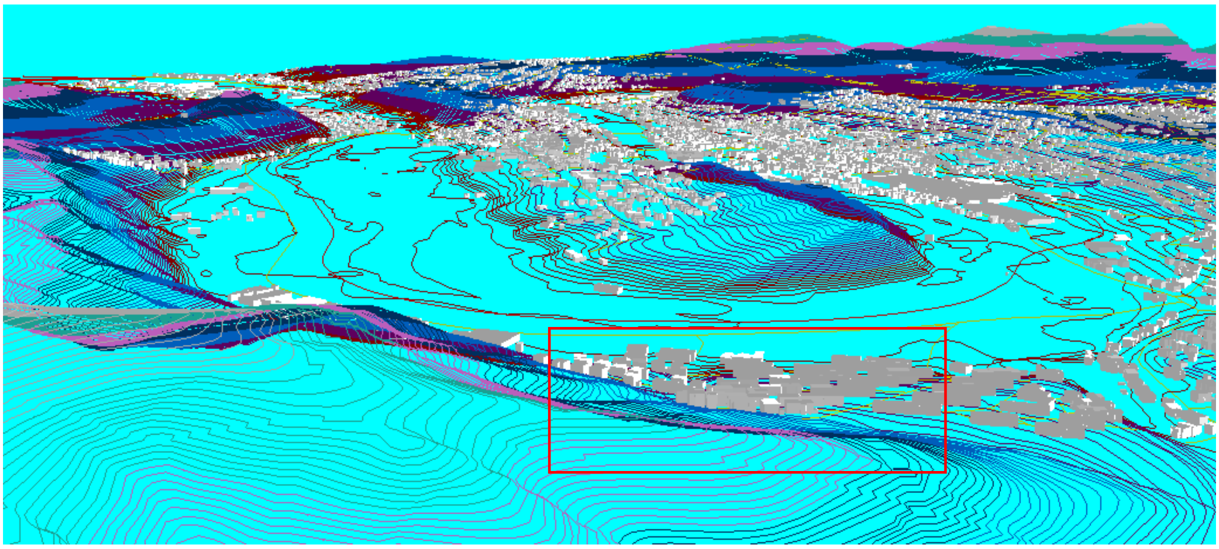


Fig. 67: 3D geodata in CityGML for the calculation of the noise map in Fig. 64: Derived contour lines for the generation of CityGML breaklines, 3D block model in CityGML, 3D road and railway data in CityGML, state road data for higher-level roads in CityGML (source: Surveying and Mapping Agency NRW, State Road Enterprise NRW, Stapelfeldt GmbH, Institute of Geodesy and Geoinformation Uni Bonn).

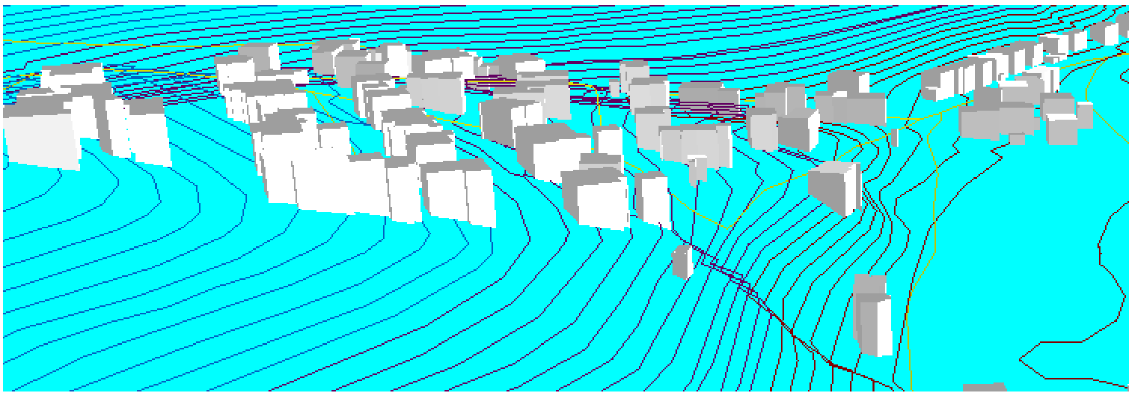


Fig. 68: Extract from Fig. 67 shows the integration of 3D block models in the DTM by appropriate CityGML modelling (lowest point of ALK building polygon is taken as measure to generate the building bottom side) (source: Surveying and Mapping Agency NRW, Stapelfeldt GmbH, Institute of Geodesy and Geoinformation Uni Bonn).

G.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.

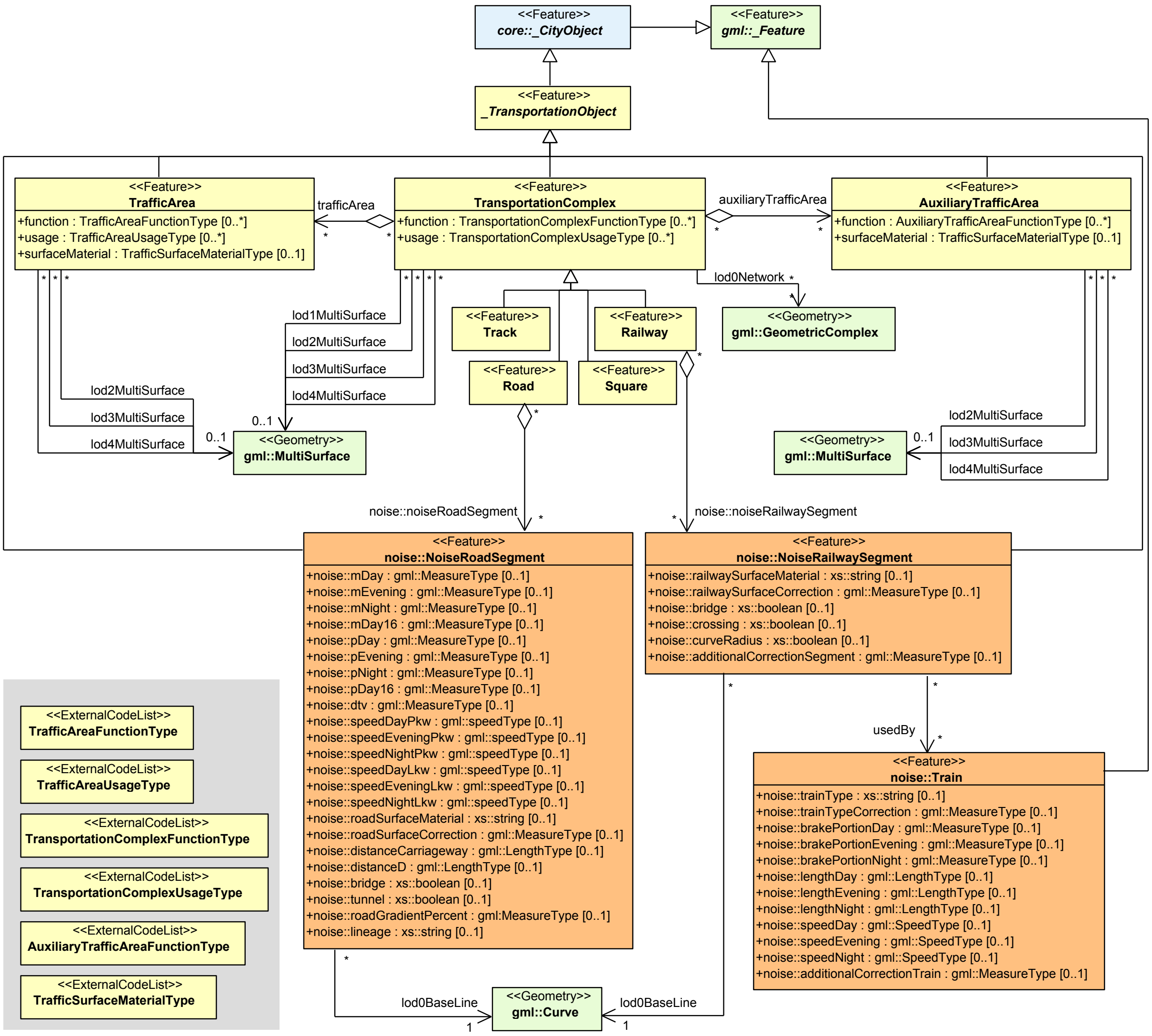


Fig. 69: CityGML noise application schema – transportation model (light yellow=CityGML *Transportation* module, light orange=CityGML Noise ADE). Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Transportation* module. The prefix *noise* is associated with the CityGML Noise ADE (source: Institute of Geodesy and Geoinformation Uni Bonn).

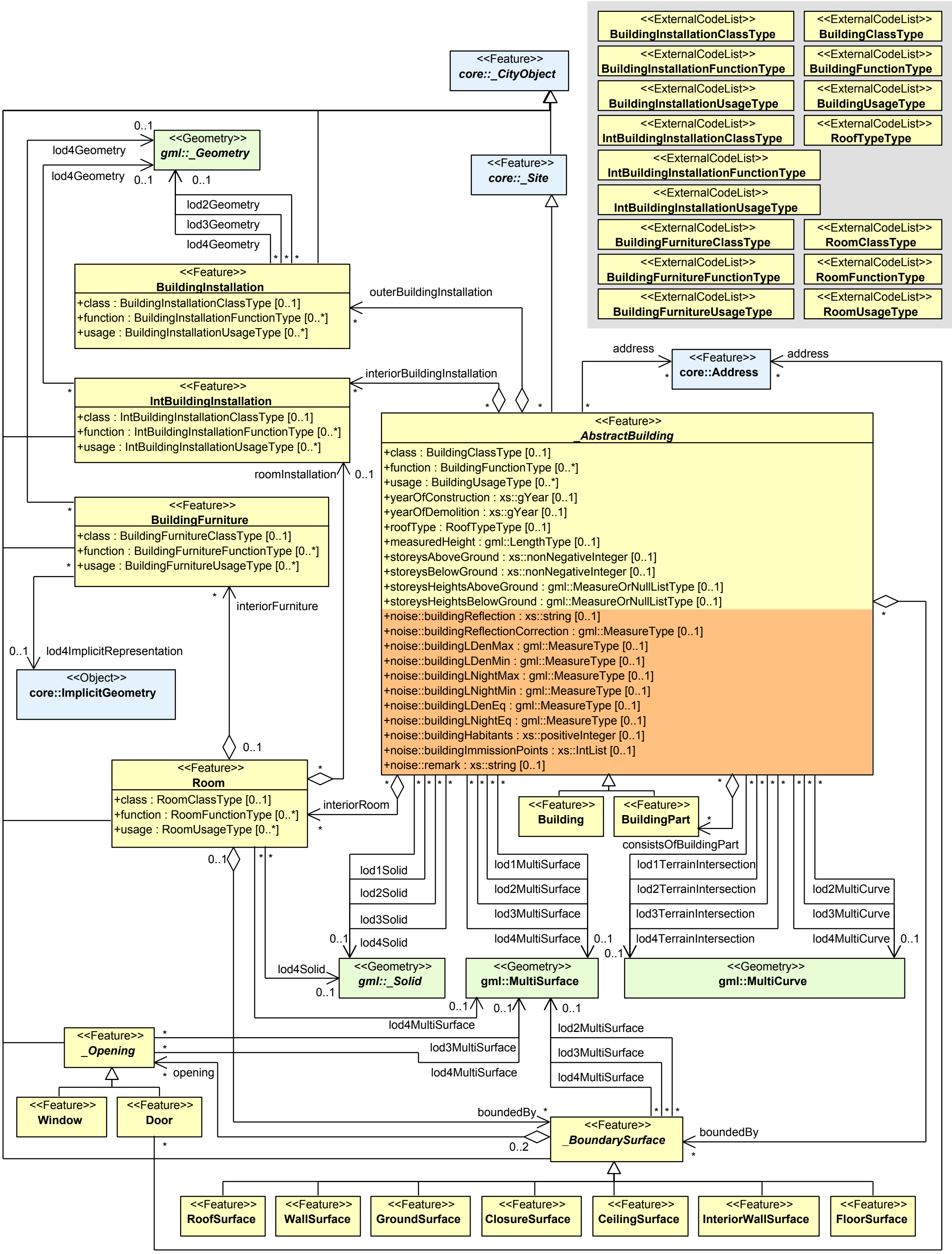


Fig. 70: CityGML noise application schema – building model (light yellow=CityGML *Building* module, light orange=CityGML Noise ADE). Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Building* module. The prefix *noise* is associated with the CityGML Noise ADE (source: Institute of Geodesy and Geoinformation Uni Bonn).

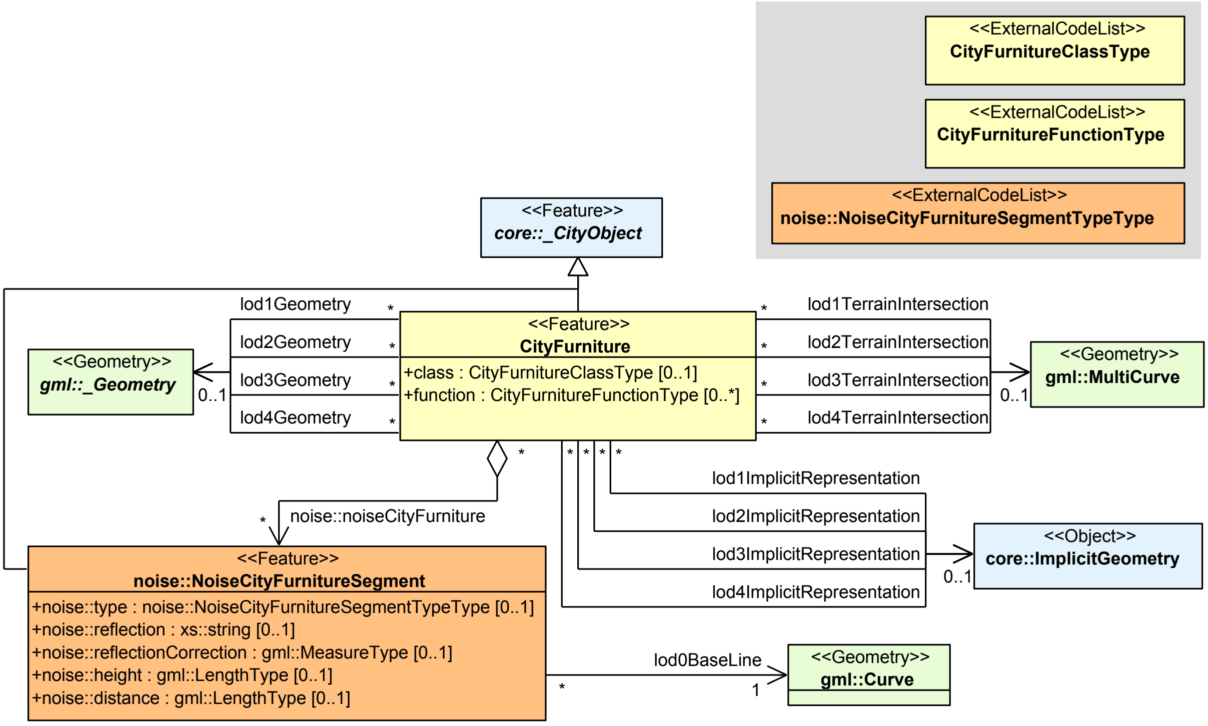


Fig. 71: CityGML noise application schema – city furniture model (light yellow=CityGML *CityFurniture* module, light orange=CityGML Noise ADE). Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *CityFurniture* module. 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

```
<xsd:schema xmlns="http://www.citygml.org/ade/noise_de" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:core="http://www.opengis.net/citygml/1.0"
xmlns:frn="http://www.opengis.net/citygml/cityfurniture/1.0"
xmlns:tran="http://www.opengis.net/citygml/transportation/1.0"
xmlns:bldg="http://www.opengis.net/citygml/building/1.0"
xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.citygml.org/ade/noise_de" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/1.0"
            schemaLocation="http://www.citygml.org/citygml/1.0/cityGMLBase.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/transportation/1.0"
            schemaLocation="http://www.citygml.org/citygml/transportation/1.0/transportation.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/building/1.0"
            schemaLocation="http://www.citygml.org/citygml/building/1.0/building.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/cityfurniture/1.0"
            schemaLocation="http://www.citygml.org/citygml/cityfurniture/1.0/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="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:simpleType name="NoiseCityFurnitureSegmentTypeType">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<!-- =========================================================================================== -->
<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:element name="noiseRailwaySegmentProperty" type="NoiseRailwaySegmentPropertyType"
             substitutionGroup="tran:_GenericApplicationPropertyOfRailway"/>
<!-- =========================================================================================== -->
<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, Train

```
<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"/>
```

G.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/1.0"
  xmlns:tran="http://www.opengis.net/citygml/transportation/1.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/1.0"
  xmlns:noise="http://www.citygml.org/ade/noise_de"
  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 http://www.citygml.org/ade/noise_de/CityGML-NoiseADE-0-5-0.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>
```



```xml
          <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 5: Excerpt from a CityGML dataset implementing the illustrated CityGML noise application schema.

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.

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.

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).

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 (Hg.): Proceedings, InterCarto – InterGIS 12. Berlin 2006.

Czerwinski, A., Kolbe, T. H., Plümer, L., Stöcker-Meier, E. (2006): Spatial data infrastructure techniques for flexible noise mapping strategies, In: Tochtermann, K. / Scharl, A. (Hg.): Proceedings of the 20th International Conference on Environmental Informatics - Managing Environmental Knowledge. Graz 2006.

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. (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., 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.

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.

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.) (2005): GML 3.1.1 Simple Dictionary Profile, Version: 1.0.0, OpenGIS® Implementation Specification, OGC Doc. No. 05-099r2.

