Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: <2020-05-06>

External identifier of this OGC® document: http://www.opengis.net/doc/IS/CityGML/3.0

Internal reference number of this OGC® document: 20-010

Version: 0.6

Category: OGC® Conceptual Model

Editors: Thomas H. Kolbe, Tatjana Kutzner, Carl Stephen Smyth, Claus Nagel, Charles Heazel, and possible further editors

OGC City Geography Markup Language (CityGML) Conceptual Model Standard

Warning

This document is not an OGC Standard. This document is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type:	OGC® Standard
Document subtype:	Conceptual Model
Document stage:	Draft
Document language:	English

License Agreement

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

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

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

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

Table of Contents

i. Abstract

CityGML is an open conceptual data model for the storage and exchange of virtual 3D city models. It is defined through a Unified Modeling Language (UML) object model. This UML model extends the ISO Technical Committee 211 (TC211) conceptual model standards for spatial and temporal data. Building on the ISO fundation assures that the man-made features described in the City Models share the same spatial-temporal universe as the surrounding countryside within which they reside.

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

ii. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, CityGML, 3D city models

iii. Preface

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

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.

More content to be provided.

iv. Submitting organizations

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

NOTE	PNG interlace method not supported for PDF generation. Find a more compatible OGC logo for "image::images/OGC_Logo.png[]"

This Document was submitted to the Open Geospatial Consortium (OGC) by the members of the CityGML Standards Working Group of the OGC. Amongst others, this comprises the following organizations:

- To be provided
- ...

v. Submitters

All questions regarding this submission should be directed to the editors or the submitters:

Table 1. Submission Contact Points

Name	Institution	Email
To be provided		

vi. Participants in development

Table 2. Participants in Development

Name	Institution
To be provided	

The table only lists persons that were involved in the development of CityGML 3.0. CityGML 3.0 is based on extensive previous work that was done for CityGML 2.0. For persons involved in the previous work, please refer to the CityGML 2.0 specification.

vii. Acknowledgements

The editors wish to thank the Special Interest Group 3D (SIG 3D) of the initiative Geodata Infrastructure Germany (GDI-DE) which originally started the development of CityGML, the CityGML Standards Working Group and the 3D Information Management (3DIM) Working Group of the OGC as well as all contributors of change requests and comments.

NOTE Explanatory text edit 01 for 01.04.2020 release starts here

Chapter 1. Introduction

1.1. 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 pedes-trian 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 sustain-able 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 securi-ty, real estate management, vehicle and pedestrian navigation, and training simulators.

CityGML is an open conceptual data model for the storage and exchange of virtual 3D city models. It is defined through a Unified Modeling Language (UML) object model. This UML model extends the ISO Technical Committee 211 (TC211) conceptual model standards for spatial and temporal data. Building on the ISO fundation assures that the man-made features described in the City Models share the same spatial-temporal universe as the surrounding countryside within which they reside.

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 differenti-ations can be represented, CityGML enables lossless information exchange between different GI systems and users.

Chapter 2. Scope

NOTE	The following text is based on Version 2.0. It will be updated

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

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

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

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

Features of CityGML:

- Geospatial information model (ontology) for urban landscapes based on the ISO 191xx family
- GML3 representation of 3D geometries, based on the ISO 19107 model
- Representation of object surface characteristics (e.g. textures, materials)
- Taxonomies and aggregations
    - Digital Terrain Models as a combination of (including nested) triangulated irregular networks (TINs), regular rasters, break and skeleton lines, mass points
    - Sites (currently buildings, bridges, and tunnels)
    - Vegetation (areas, volumes, and solitary objects with vegetation classification)
    - Water bodies (volumes, surfaces)
    - Transportation facilities (both graph structures and 3D surface data)
    - Land use (representation of areas of the earth’s surface dedicated to a specific land use)
    - City furniture
    - Generic city objects and attributes
    - User-definable (recursive) grouping
- Multiscale model with 5 well-defined consecutive Levels of Detail (LOD):
    - LOD0 – regional, landscape

    - LOD1 – city, region
    - LOD2 – city districts, projects
    - LOD3 – architectural models landmarks
- 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 ap-plication 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.

NOTE	End of text to be updated

Chapter 3. Conformance

NOTE

Update to reflect change in approach to requirements

This standard defines a Conceptual Model which is independent of any encoding or formatting techniques. The Standardization Targets for this standard are Implementation Specifications. These Implementation Specifications specify how the Conceptual Model shall be implemented for specific technologies. Conformant Implementation Specifications provide evidence that they are an accurate representation of the Conceptual Model. This evidence shall include implementations of the abstract tests specified in Annex A (normative) of this document.

Since this standard is agnostic to the implementating technologies, the specific techniques to be used for conformance testing cannot be specified. It is the responsiblity of the Implementation Specifications to provide evidence of conformance which is appropriate for the implementing technologies. This evidence should be provided as an annex to the Implementation Specification document.

This standard identifies seventeen (17) conformance classes. One conformance class is defined for each Package in the UML model. Each conformance class is defined by one requirements class. The tests in Annex A are organized by Requirements Class. So an implementation of the *Core* conformance class must pass all tests specified in Annex A for the *Core* requirements class.

Of these seventeen conformance classes, Implementation Specifications are only required to implement the *Core* conformance class. All other conformance classes are optional. In the case where an implemented conformance class has a dependency on another conformance class, that conformance class shall also be implemented.

The CityGML Conceptual Model is defined by the CityGML UML model. This specification is a representation of that UML model in document form. In the case of a discrepency between the UML model and this document, the UML model is authoritative.

Chapter 4. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of OGC 20-010. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of OGC 12-019 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

- IETF: RFC 2045 & 2046, Multipurpose Internet Mail Extensions (MIME). (November 1996)
- IETF: RFC 3986, Uniform Resource Identifier (URI): Generic Syntax. (January 2005)
- ISO: ISO 8601:2004, Data elements and interchange formats – Information interchange – Representation of dates and times
- ISO: ISO 19103:2015, Geographic Information – Conceptual Schema Language
- ISO: ISO 19105:2000, Geographic information – Conformance and testing
- ISO: ISO 19107:2003, Geographic Information – Spatial Schema
- ISO: ISO 19109:2015, Geographic Information – Rules for Application Schemas
- ISO: ISO 19111:2019, Geographic information – Referencing by coordinates
- ISO: ISO 19115-1:2014, Geographic information — Metadata — Part 1: Fundamentals
- ISO: ISO 19123:2005, Geographic information — Schema for coverage geometry and functions
- ISO: ISO 19156:2011, Geographic information – Observations and measurements
- ISO: ISO/IEC 19505-2:2012, Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure
- ISO/IEC 19507:2012, Information technology — Object Management Group Object Constraint Language (OCL)
- ISO: ISO/IEC 19775-1:2013 Information technology — Computer graphics, image processing and environmental data representation — Extensible 3D (X3D) — Part 1: Architecture and base components
- Khronos Group Inc.: COLLADA – Digital Asset Schema Release 1.5.0
- OASIS: extensible Address Language (xAL v2.0)
- OGC: The OpenGIS® Abstract Specification Topic 5: Features, OGC document 08-126
- OGC: The OpenGIS™ Abstract Specification Topic 8: Relationships Between Features, OGC document 99-108r2
- OGC: The OpenGIS™ Abstract Specification Topic 10: Feature Collections, OGC document 99-110

Chapter 5. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r8], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this Best Practice.

For the purposes of this document, the following additional terms and definitions apply.

2D data
geometry of features is represented in a two-dimensional space
NOTE In other words, the geometry of 2D data is given using (X,Y) coordinates.
[INSPIRE D2.8.III.2, definition 1]

2.5D data
geometry of features is represented in a three-dimensional space with the constraint that, for each (X,Y) position, there is only one Z
[INSPIRE D2.8.III.2, definition 2]

3D data
Geometry of features is represented in a three-dimensional space.
NOTE In other words, the geometry of 2D data is given using (X,Y,Z) coordinates without any constraints.
[INSPIRE D2.8.III.2, definition 3]

conceptual model
model that defines concepts of a universe of discourse
[ISO 19101-1:2014, 4.1.5]

conceptual schema
formal description of a conceptual model
[ISO 19101-1:2014, 4.1.6]

conformance test class
set of conformance test modules that must be applied to receive a single certificate of conformance
[OGC 08-131r3, definition 4.4]

feature
abstraction of real world phenomena
[ISO 19101-1:2014, definition 4.1.11]

feature attribute
characteristic of a feature
[ISO 19101-1:2014, definition 4.1.12]

feature type
class of features having common characteristics
[ISO 19156:2011, definition 4.7]

level of detail

quantity of information that portrays the real world
NOTE The concept comprises data capturing rules of spatial object types, the accuracy and the types of geometries, and other aspects of a data specification. In particular, it is related to the notions of scale and resolution.
[INSPIRE Glossary]

life-cycle information
set of properties of a spatial object that describe the temporal characteristics of a version of a spatial object or the changes between versions
[INSPIRE Glossary]

measurement
set of operations having the object of determining the value of a quantity
[ISO 19101-2:2018, definition 3.21] / [VIM:1993, 2.1]

model
abstraction of some aspects of reality
[ISO 19109:2015, definition 4.15]

observation
act of measuring or otherwise determining the value of a property
[ISO 19156:2011, definition 4.11]

observation procedure
method, algorithm or instrument, or system of these, which may be used in making an observation
[ISO 19156:2011, 4.12]

observation result
estimate of the value of a property determined through a known observation procedure
[ISO 19156:2011, 4.14]

property
facet or attribute of an object referenced by a name.
[ISO 19143:2010, definition 4.21]

requirements class
aggregate of all requirement modules that must all be satisfied to satisfy a conformance test class
[OGC 08-131r3, definition 4.19]

schema
formal description of a model
[ISO 19101-1:2014, definition 4.1.34]

sensor
type of observation procedure that provides the estimated value of an observed property at its output
[OGC 08-094r1, definition 4.5]

timeseries
sequence of data values which are ordered in time

[OGC 15-043r3]

universe of discourse
view of the real or hypothetical world that includes everything of interest
[ISO 19101-1:2014, definition 4.1.38]

version
Particular variation of a spatial object
[INSPIRE Glossary]

5.1. Abbreviated Terms

The following abbreviated terms are used in this document:

The list of acronyms needs to be reviewed once all sections have been updated.

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

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

Chapter 6. Conventions

6.1. Identifiers

The normative provisions in this document are denoted by the URI

http://www.opengis.net/spec/CityGML/3.0

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

6.2. 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 Figure 1.

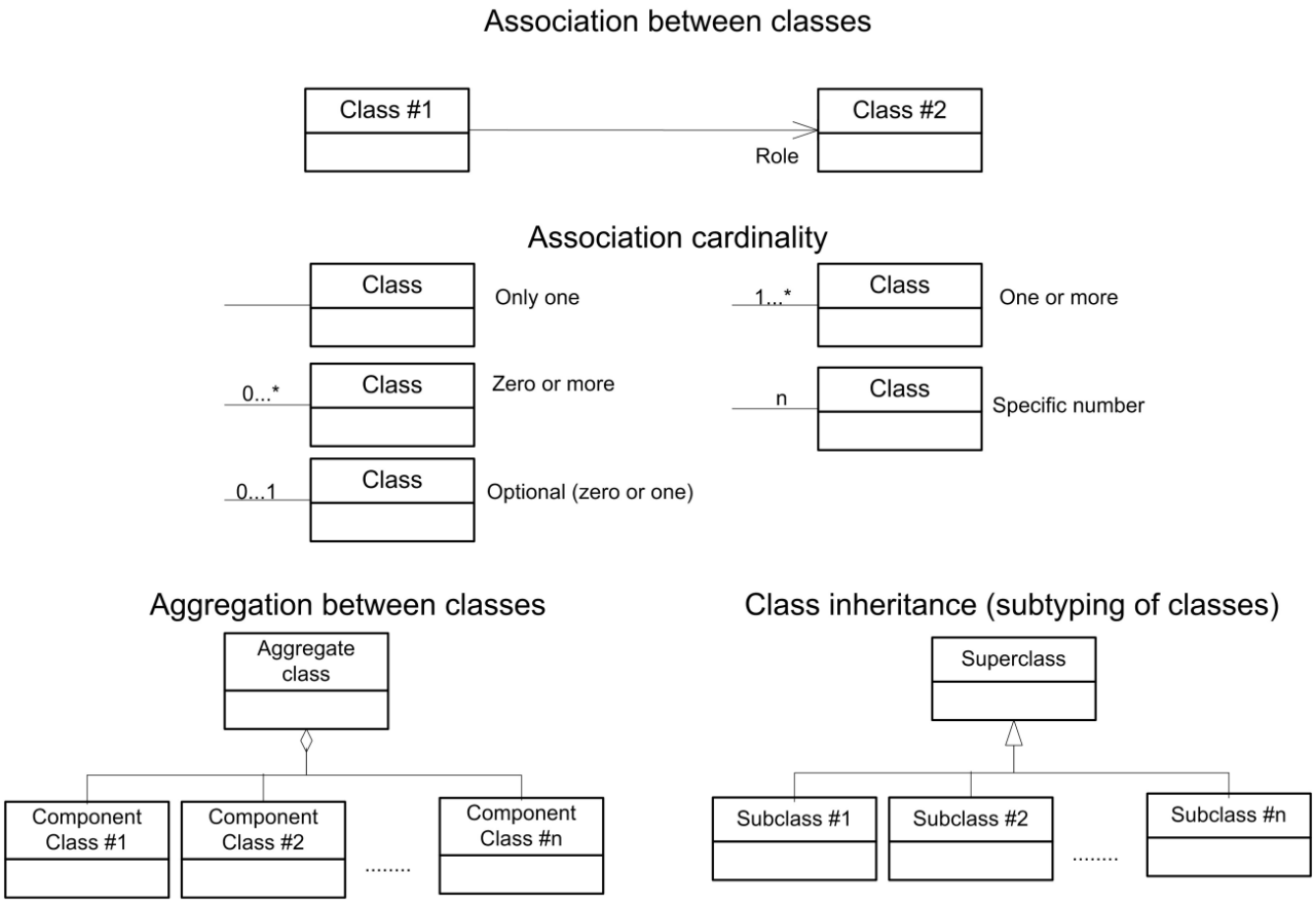

Figure 1. UML notation (see ISO TS 19103, Geographic information - Conceptual schema language).

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.

NOTE Need formal definitions for the sterotypes

The following stereotypes are used:

- <<FeatureType>>
- <<TopLevelFeatureType>>
- <<type>>
- <<ObjectType>>
- <<DataType>>is used as a descriptor of a set of values that lack identity. Data types include primitive prede-fined 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.
- <<enumeration>>enumerates the valid attribute values in a fixed list of named literal values. Enumerations are specified in the CityGML schema.
- <<BasicType>>
- <<CodeList>>enumerates the valid attribute values. In contrast to Enumeration, the list of values is open and, thus, not given inline the CityGML UML Model. The allowed values can be provided within an external code list.
- <<Union>>is a list of attributes. The semantics are that only one of the attributes can be present at any time.

NOTE Update to the current convention

In order to enhance the readability of the CityGML UML diagrams, classes are depicted in different colors if they belong to different UML packages. The following coloring scheme is applied:

Feature Types: Yellow TopLevelFeatureType: Orange Type :Green ObjectType: Orange DataType: Blue Enumration: Purple Basictype: Brown Codelist: Pink Union: Blue

- Classes painted in yellow belong to the UML package which is subject of discussion in that clause of the specification in which the UML diagram is given. For example, in the context of chapter 10.1 which introduces the *CityGML Core* module, the yellow color is used to denote classes which are defined in the *CityGML Core* UML package. Likewise, the yellow classes shown in UML diagrams in chapter 10.3 are associated with the *Building* module which is subject of discussion in that chapter.
- Classes painted in blue belong to a CityGML UML package different to that associated with the yellow color. In order to explicitly denote the UML package of such classes, their class names carry a naoughout this specification mespace prefix which is uniquely associated with a CityGML module throughout this specification . For example, in the context of the *Building* module, classes from the *CityGML Core* module are painted in blue and their class names are preceded by the prefix core.

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

geometry element defined by GML3.

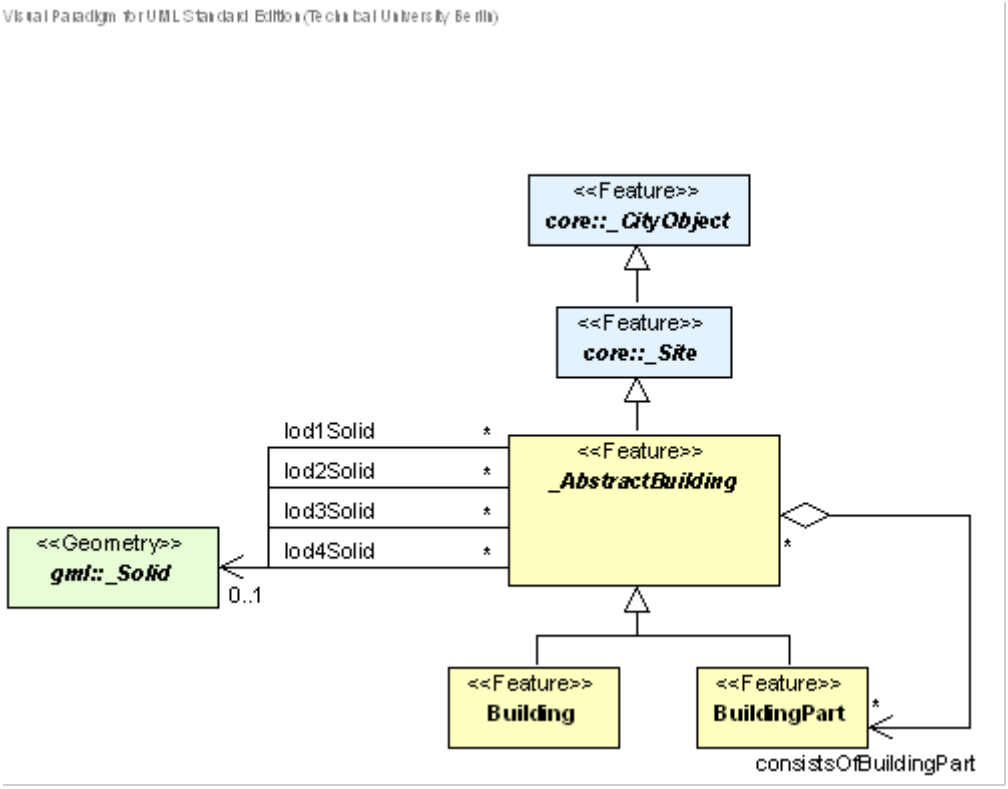

Figure 2. Example UML diagram demonstrating the UML notation and coloring scheme used throughout the CityGML specification.

6.3. Conceptual Modeling (Informative)

ISO 19101 defines universe of discourse to be a view of the real or hypothetical world that includes everything of interest. That standard then defines conceptual model to be a model that defines concepts of a universe of discourse.

The scope of this CityGML Conceptual Model Standard establishes the limits of the universe of discourse for this Standard. The next task is to discover and standardize the concepts within this scope. CityGML will potentially support numerous diverse application software packages covering multiple disciplines and facility life cycle phases. Each conceivably can have its own universe of discourse and their own set of concepts.

The goal of this CityGML Conceptual Model Standard is to establish and document a common set of concepts that spans the applications supported. This does not attempt to redefine application concepts, but merely present a common set of concepts from and to which their concepts can be understood and mapped.

GML and JSON encodings are planned and other encodings are anticipated. Each encoding addresses a specific information community and set of application software packages. However, with the increasing desire to share information between communities and applications having a common conceptual model across all of these encodings is highly advantageous.

An added benefit of the development of a conceptual model results from the rigor involved in achieving consensus. After numerous iterations, the end result is consistent, cohesive, and complete. Updating a conceptual model is far easier than rewriting software code. Further, the

iterations help to flesh out details as well as to unearth differences in individual conceptualizations.

Perhaps the greatest benefit of the standards activity is the ability to communicate the resultant model. This is in part due to using a standardized conceptual modelling language like UML and the agreed OGC and TC211 conventions for using UML. The eventual outcome of being able to provide formal documentation for what is meant by each concept is invaluable in understanding the subsequent encodings and applications.

This will be the first OGC conceptual model standard, without accompanying encodings. Yet the model is presented in a manner consistent with the formalisms adopted for writing OGC standards. This standard follows the OGC Specification Model standard for modular specifications [10] and is consistent with the OGC Naming Authority conventions and recommendations. The target of this Standard are the encoding standards which will follow and not the application software that will implement these encodings. Requirements for the encodings are explicit and grouped into Requirements Classes. Accompanying Conformance Classes are included to determine if an encoding conforms to the conceptual model.

UML has been used as the conceptual modeling language in this Standard. Class Diagrams have been created and inserted as Figures. The boxes in these diagrams (officially “Classifiers” in UML) typically represent classes, data types, enumerations, code lists, unions, etc. and this terminology is used throughout the Standard. However, since this is a Conceptual Model, these should all be interpreted to be “concepts”. For each Requirements Class, an introductory diagram is included which contains all of the concepts relevant to that Requirements Class. However, the boxes are simplified by suppressing attributes. These attributes are provided in a series of context diagrams which follow, each focusing on a particular set of concepts in the Requirements Class.

Though redundant with the UML diagrams, all of the class attributes are repeated in the document text, including attribute definitions not visible in the diagrams. If these differ, the UML takes precedence. Because association roles behave similar to attributes, they appear at the end of the textual attribute listing as if they were attributes. The cardinality of the association is depicted as the attribute cardinality and the associated class as the data type.

Chapter 7. 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 interna-tional 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 mainte-nance 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 representa-tion of the semantic and thematic properties, taxonomies and aggregations. CityGML includes a geometry model and a thematic model. The geometry model allows for the consistent and homogeneous definition of geometrical and topological properties of spatial objects within 3D city models (chapter 8). The base class of all objects is _ *CityObject* which is a subclass of the GML class _ *Feature*. All objects inherit the properties from _ *CityObject*.

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

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

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

NOTE Explanatory text edit 06 for 01.04.2020 release ends here

Chapter 8. CityGML UML Model

The CityGML UML model is the normative definition of the CityGML Conceptual Model. The tables and figures in this section were software generated from the UML model. As such, this section provides a normative representation of the CityGML Conceptual Model.

An alternate representation can be found in the CityGML Data Dictionary.

8.1. Structural Overview of Requirements Classes

The Requirements Classes for this standard are structured as UML Packages as illustrated in CityGML UML Packages. Each Requirements Class is specified in detail in their respective subsections. These subsections include a UML diagram, data dictionary, and the applicable requirements.

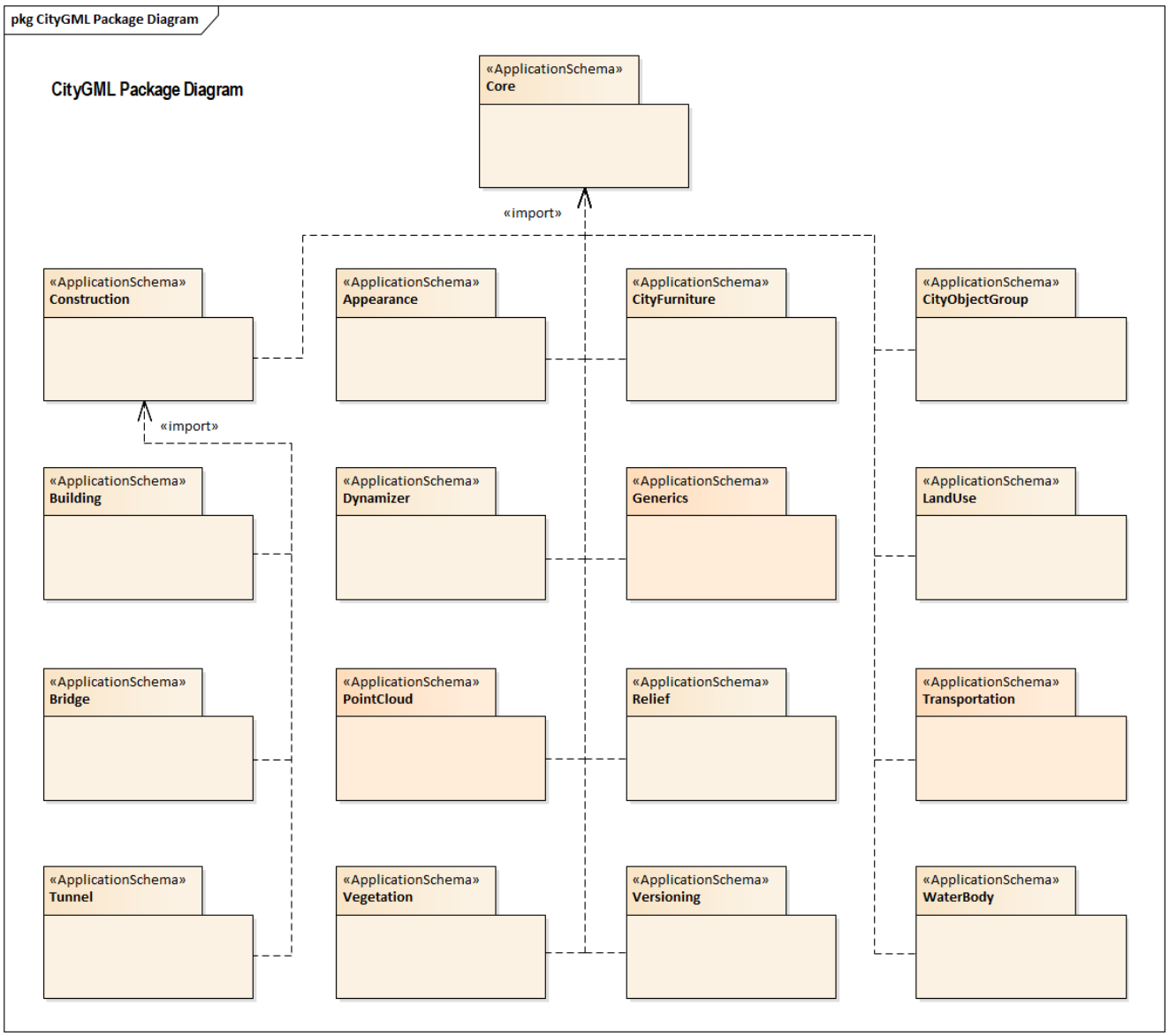

Figure 3. CityGML UML Packages

8.2. Core

Requirements Class	
http://www.opengis.net/spec/CityGML/3.0/req/req-class-core	
Target type	Implementation Specification
Dependency	ISO 19103:2015
Dependency	ISO 19107:2003
Dependency	ISO 19109:2015
Dependency	ISO 19111:2019
Dependency	ISO 19123:2005

The Core module defines the basic concepts and components of city models. This rather large body of work has been divided into five sections. These sections build on each other from the fundamental principles specified by the ISO up through the full CityGML model. These sections are summarized in the CityGML Core Sections table.

Table 3. CityGML Core Sections

The Use of ISO Standards	Describes the use of ISO 191** standards to provide a foundation to the CityGML model.
Space Concepts	Defines the concepts of space as used in the CityGML model
Geometry and LOD	Defines the geometry and Level Of Detail concepts
CityGML	The core elemtents of the model. where it all comes together
Types, Enumerations and Codelist	Defines the little things which make this model work.

8.2.1. Requirements

Requirement 1	/req/Core/Classes
For each UML class defined or referenced in the Core Package:	
A	The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
B	The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and cardinalities as those of the UML class.

C	The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and cardinality.
D	The implementation Specification SHALL represent the attributes of all parent classes of the the UML class including the name, definition, type, and cardinality.

8.2.2. ISO Dependencies

CityGML builds on the ISO 191** family of standards. The applicable standards are identified in the Use of ISO Standards in CityGML diagram. Data dictionaries are included for all of the ISO-defined classes explicitly referenced in the CityGML UML model. These data dictionaries are provided for the convinence of the user. The ISO standards are the normative source.

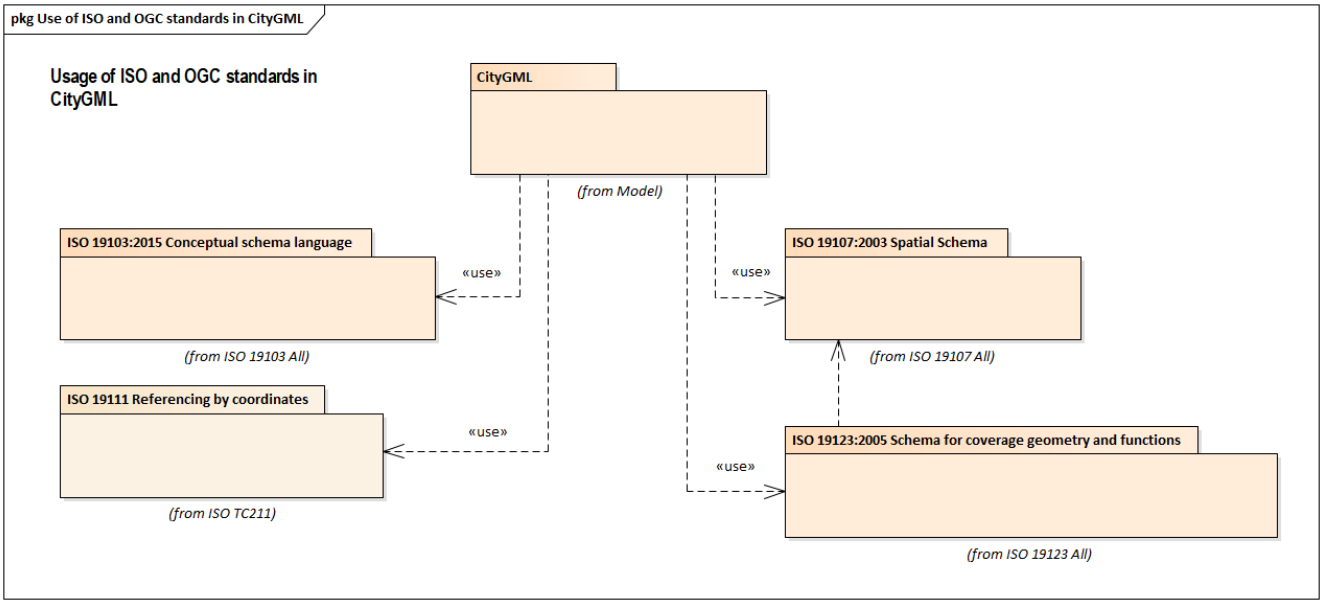

Figure 4. Use of ISO Standards in CityGML

The ISO classes explicitly used in the CityGML UML model are introduced in the ISO Classes used in CityGML table. More detail about these classes can be found in the Data Dictionaries section.

Table 4. ISO Classes used in CityGML

Class Name	Description
AnyFeature	A generalization of all feature types
CV_DiscreteGridPointCoverage	undefined
GM_Object	root class of the geometric object taxonomy.
GM_MultiCurve	undefined.
GM_MultiSurface	undefined.
GM_Point	The basic data type for a geometric object consisting of one and only one point.

GM_Solid	The basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces.
GM_Surface	The basis for 2-dimensional geometry.
GM_Tin	A GM_TriangulatedSurface which uses the Delaunay or similar algorithm.
GM_TriangulatedSurface	A GM_PolyhedralSurface that is composed only of triangles
SC_CRS	Coordinate reference system which is usually single but may be compound.

8.2.3. Spatial Concepts

All city objects are differentiated into spaces and space boundaries. Spaces are entities of volumetric extent in the real world. Buildings, water bodies, trees, rooms, and traffic spaces, for instance, have a volumetric extent. Spaces can be classified into physical spaces and logical spaces. Physical spaces, in turn, can be further classified into occupied spaces and unoccupied spaces.

Space boundaries, in contrast, are entities with a real extent in the real world. Space boundaries can be differentiated into different types of thematic surfaces, such as wall surfaces and roof surfaces.

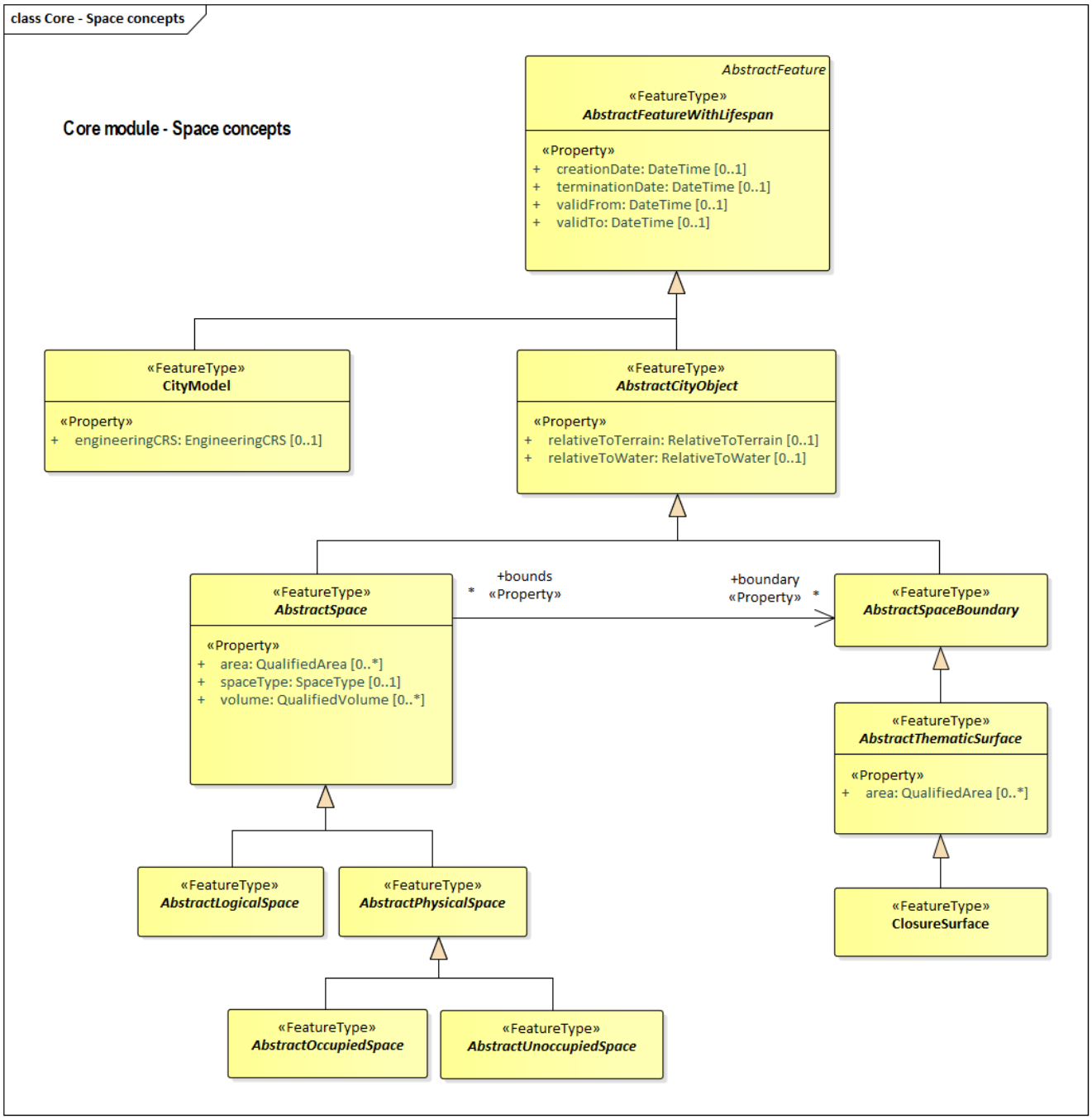

Figure 5. UML Space Concepts

The Space Concept classes defined in the CityGML UML model are introduced in the Spatial Classes used in Core table. More detail about these classes can be found in the Data Dictionaries section.

Table 5. Spatial Classes used in Core

Class	Description
AbstractCityObject «FeatureType»	AbstractCityObject is the abstract superclass of all thematic classes within the CityGML data model.
AbstractFeatureWithLifespan «FeatureType»	AbstractFeatureWithLifespan is the base class for all CityGML features. It allows the optional specification of the real-world and database times for the existence of each feature.

AbstractLogicalSpace «FeatureType»	AbstractLogicalSpace is the abstract superclass for all types of logical spaces. Logical space refers to spaces that are not bounded by physical surfaces but are defined according to thematic considerations.
AbstractOccupiedSpace «FeatureType»	AbstractOccupiedSpace is the abstract superclass for all types of physically occupied spaces. Occupied space refers to spaces that are partially or entirely filled with matter.
AbstractPhysicalSpace «FeatureType»	AbstractPhysicalSpace is the abstract superclass for all types of physical spaces. Physical space refers to spaces that are fully or partially bounded by physical objects.
AbstractSpace «FeatureType»	AbstractSpace is the abstract superclass for all types of spaces. A space is an entity of volumetric extent in the real world.
AbstractSpaceBoundary «FeatureType»	AbstractSpaceBoundary is the abstract superclass for all types of space boundaries. A space boundary is an entity with areal extent in the real world. Space boundaries are objects that bound a Space. They also realize the contact between adjacent spaces.
AbstractThematicSurface «FeatureType»	AbstractThematicSurface is the abstract superclass for all types of thematic surfaces.
AbstractUnoccupiedSpace «FeatureType»	AbstractUnoccupiedSpace is the abstract superclass for all types of physically unoccupied spaces. Unoccupied space refers to spaces that are entirely or mostly free of matter.
CityModel «FeatureType»	CityModel is the container for all objects belonging to a city model.
ClosureSurface «FeatureType»	ClosureSurface is a special type of thematic surface used to close holes in volumetric objects. Closure surfaces are virtual (non-physical) surfaces.

8.2.4. Geometry and LOD

Spaces and space boundaries can have various geometry representations depending on the Level of Detail (LOD). Spaces can be spatially represented as single points in LOD0, multi-surfaces in LOD0/2/3, solids in LOD1/2/3, and multi-curves in LOD2/3. Space boundaries can be represented as multi-surfaces in LOD0/2/3 and as multi-curves in LOD2/3. All Levels of Detail allow for the representation of the interior of city objects.

In addition, the geometry can also be represented implicitly, i.e. the shape is stored only once as a prototypical geometry, which then is re-used or referenced, wherever the corresponding feature occurs in the 3D city model.

The thematic classes, such as building, tunnel, road, land use, water body, or city furniture are defined as subclasses of the space and space boundary classes within the thematic modules. Since all city objects in the thematic modules represent subclasses of the space and space boundary classes, they automatically inherit the geometries defined in the Core module.

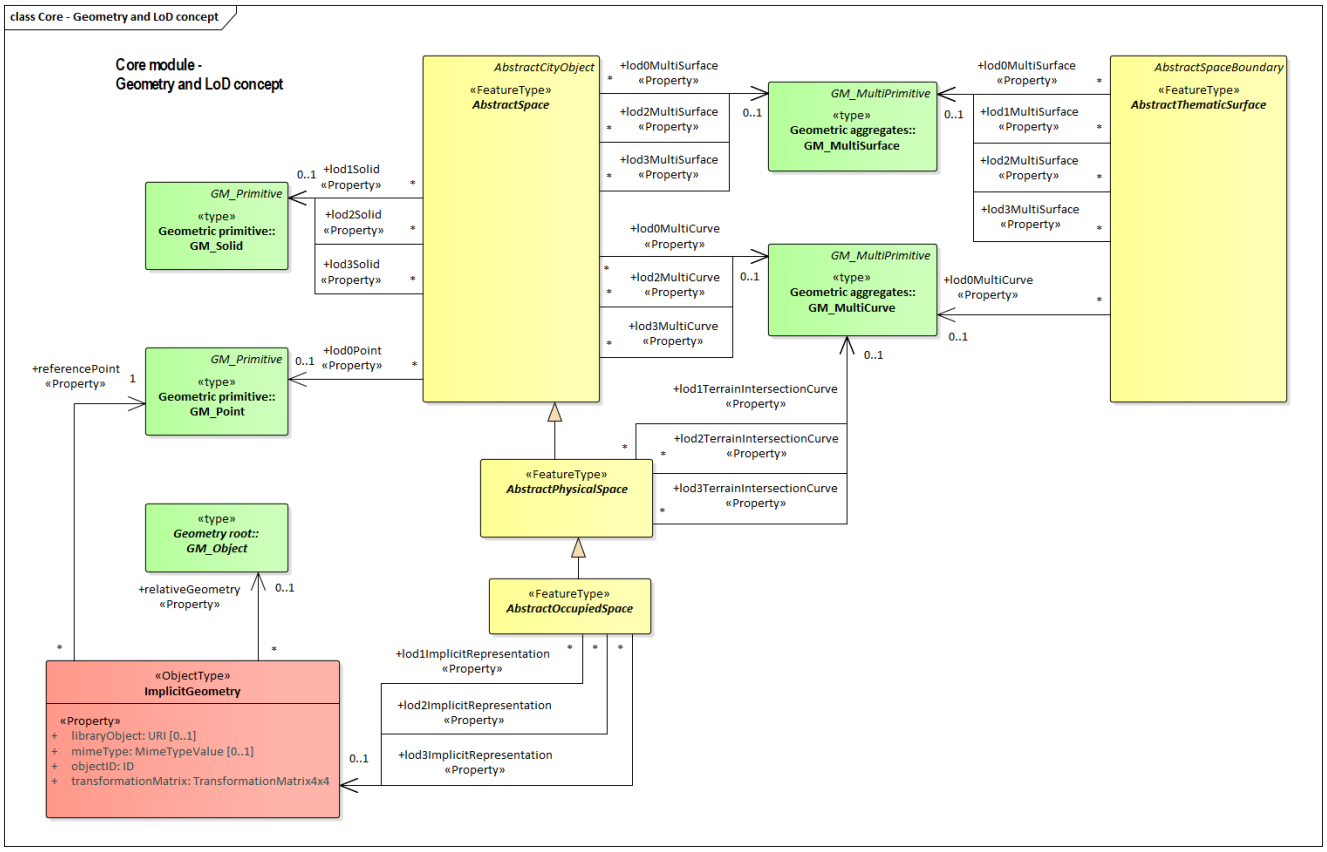

Figure 6. UML Geometry and LoD Concepts

The Geometry and LOD Concept classes defined in the CityGML UML model are introduced in the Geometry Classes used in Core table. More detail about these classes can be found in the Data Dictionaries section.

Of particular note is the Implicit Geometry concept. Many of the objects encountered in a city landscape have the same geometry. How many types of street lamps can there be? An Implicit Geometry captures that geometry once, and re-uses that one geometry for all similar street lamp objects.

Table 6. Geometry Classes used in Core

Class	Description
AbstractOccupiedSpace «FeatureType»	AbstractOccupiedSpace is the abstract superclass for all types of physically occupied spaces. Occupied space refers to spaces that are partially or entirely filled with matter.
AbstractPhysicalSpace «FeatureType»	AbstractPhysicalSpace is the abstract superclass for all types of physical spaces. Physical space refers to spaces that are fully or partially bounded by physical objects.
AbstractPointCloud «FeatureType»	AbstractPointCloud is the abstract superclass to represent PointCloud objects.
AbstractSpace «FeatureType»	AbstractSpace is the abstract superclass for all types of spaces. A space is an entity of volumetric extent in the real world.

AbstractSpaceBoundary «FeatureType»	AbstractSpaceBoundary is the abstract superclass for all types of space boundaries. A space boundary is an entity with areal extent in the real world. Space boundaries are objects that bound a Space. They also realize the contact between adjacent spaces.
AbstractThematicSurface «FeatureType»	AbstractThematicSurface is the abstract superclass for all types of thematic surfaces.
ImplicitGeometry «ObjectType»	ImplicitGeometry is a geometry representation 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 can be re-used or referenced many times, wherever the corresponding feature occurs in the 3D city model.

8.2.5. CityGML

City models are virtual representations of real-world cities and landscapes. A city model aggregates different types of objects, which can be city objects, appearances, different versions of the city model, transitions between different versions of the city model, and feature objects. All objects defined in CityGML are features with lifespan. This allows the optional specification of the real-world and database times for the existence of each feature, as is required by the Versioning module (cf. Section Versioning). Features that define thematic concepts related to cities and landscapes, such as building, bridge, water body, or land use, are referred to as city objects.

The UML diagram of the Core module is depicted in the Core UML Diagram.

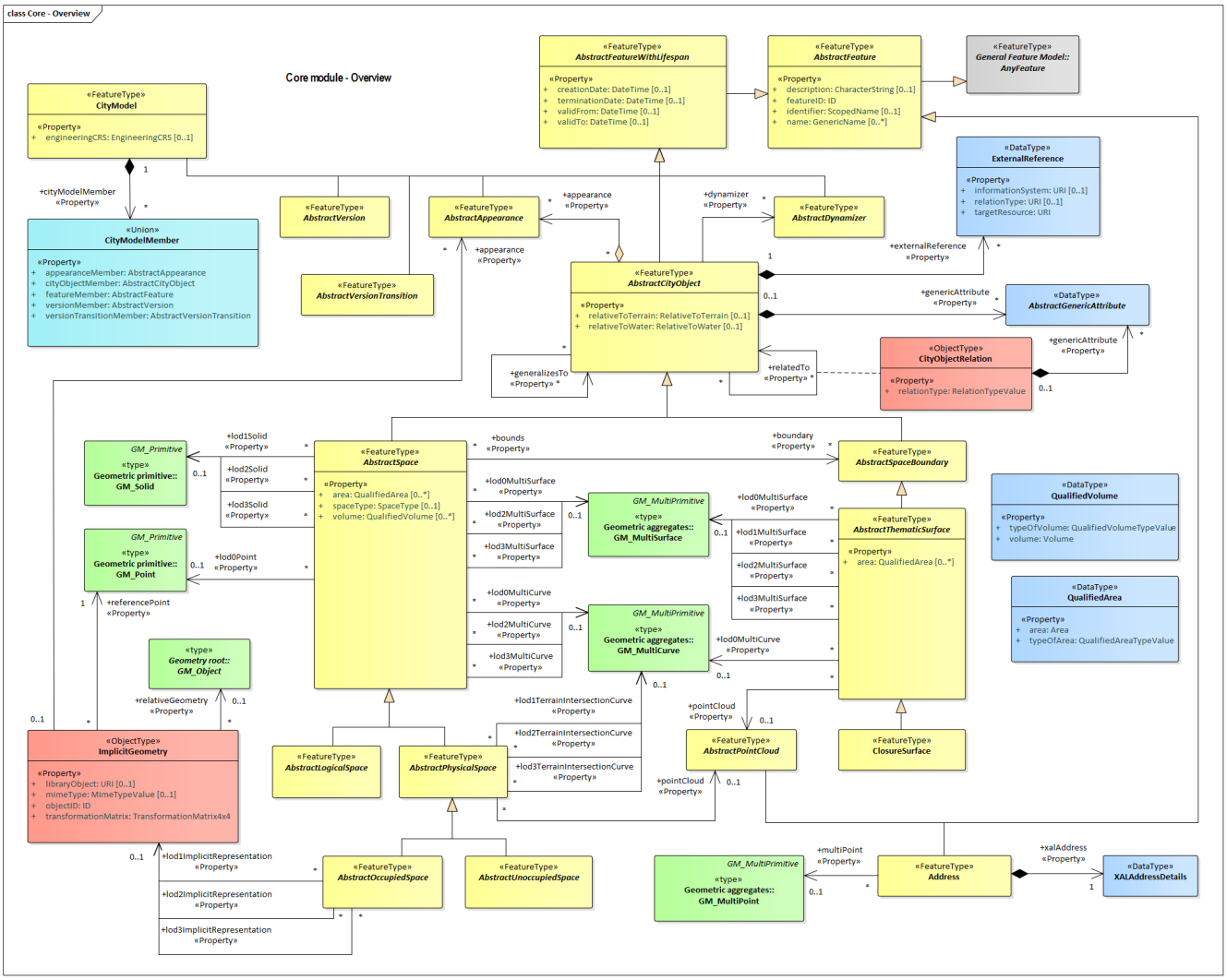

Figure 7. UML diagram of CityGML's core module.

The Geometry and LOD Concept classes define the majority of the CityGML Core. However, there are some additional classes required to fill out this section. These classes are introduced in the [core-class-table] table. More detail about these classes can be found in the Data Dictionaries section.

Table 7. Classes used in Core

Class	Description
AbstractAppearance «FeatureType»	AbstractAppearance is the abstract superclass to represent any kind of appearance objects.
AbstractDynamizer «FeatureType»	AbstractDynamizer is the abstract superclass to represent Dynamizer objects.
AbstractFeature «FeatureType»	AbstractFeature is the abstract superclass of all feature types within the CityGML data model.
AbstractVersion «FeatureType»	AbstractVersion is the abstract superclass to represent Version objects.
AbstractVersionTransition «FeatureType»	AbstractVersionTransition is the abstract superclass to represent VersionTransition objects.

Address «FeatureType»	Address represents an address of a city object.
CityObjectRelation «ObjectType»	CityObjectRelation represents a specific relation from the city object in which it is included to another city object.

8.2.6. Data types, Enumerations and CodeLists

While FeatureTypes capture the real-world concepts on the CityGML Conceptual Model, they would be incomplete without the additional concepts from which they are made. These supporting constructs are defined in the following tables.

Table 8. Data Types used in Core

Name	Description
AbstractGenericAttribute «DataType»	AbstractGenericAttribute is the abstract superclass for all types of generic attributes.
ExternalReference «DataType»	ExternalReference is a reference to a corresponding object in another information system, for example in the German cadastre (ALKIS), the German topographic information system (ATKIS), or the OS UK MasterMap®.
Occupancy «DataType»	Occupancy is an application-dependent indication of what is contained by a feature.
QualifiedArea «DataType»	QualifiedArea is an application-dependent measure of the area of a space or of a thematic surface.
QualifiedVolume «DataType»	QualifiedVolume is an application-dependent measure of the volume of a space.
XALAddressDetails «DataType»	XALAddressDetails represents address details according to the OASIS xAL standard.

Table 9. Primitive Data Types used in Core

Name	Description
DoubleBetween0and1 «BasicType»	DoubleBetween0and1 is a basic type for values, which are greater or equal than 0 and less or equal than 1. The type is used for color encoding, for example.
DoubleBetween0and1List «BasicType»	DoubleBetween0and1List is a basic type that represents a list of double values greater or equal than 0 and less or equal than 1. The type is used for color encoding, for example.
DoubleList «BasicType»	DoubleList is an ordered sequence of double values.
DoubleOrNilReasonList «BasicType»	DoubleOrNilReasonList is a basic type that represents a list of double values and/or nil reasons.

ID «BasicType»	ID is a basic type that represents a unique identifier.
IntegerBetween0and3 «BasicType»	IntegerBetween0and3 is a basic type for integer values, which are greater or equal than 0 and less or equal than 3. The type is used for encoding the LOD number.
MeasureOrNilReasonList «BasicType»	MeasureOrNilReasonList is a basic type that represents a list of double values and/or nil reasons together with a unit of measurement.
TransformationMatrix2x2 «BasicType»	TransformationMatrix2x2 is a 2 by 2 matrix represented as a list of four double values in row major order.
TransformationMatrix3x4 «BasicType»	TransformationMatrix3x4 is a 3 by 4 matrix represented as a list of twelve double values in row major order.
TransformationMatrix4x4 «BasicType»	TransformationMatrix4x4 is a 4 by 4 matrix represented as a list of sixteen double values in row major order.

Table 10. Union types used in Core

Name	Description
CityModelMember «Union»	CityModelMember is a union type that enumerates the different types of objects that can occur as members of a city model.
DoubleOrNilReason «Union»	DoubleOrNilReason is a union type that allows for choosing between a double value and a nil reason.
NilReason «Union»	NilReason is a union type that allows for choosing between two different types of nil reason.

Table 11. Enumerated Classes used in Core

Name	Description
RelativeToTerrain «Enumeration»	RelativeToTerrain enumerates the spatial relations of a city object relative to terrain in a qualitative way.
RelativeToWater «Enumeration»	RelativeToWater enumerates the spatial relations of a city object relative to the water surface in a qualitative way.
SpaceType «Enumeration»	SpaceType is an enumeration that characterises a space according to its closure properties.

Table 12. CodeList Classes used in Core

Name	Description
IntervalValue «CodeList»	IntervalValue is a code list used to specify a time period.

MimeTypeValue «CodeList»	MimeTypeValue is a code list used to specify the MIME type of a referenced resource.
NilReasonEnumeration «CodeList»	NilReasonEnumeration is a code list that enumerates the different nil reasons.
OccupantTypeValue «CodeList»	OccupantTypeValue is a code list used to classify occupants.
OtherRelationTypeValue «CodeList»	OtherRelationTypeValue is a code list used to classify other types of city object relations.
QualifiedAreaTypeValue «CodeList»	QualifiedAreaTypeValue is a code list used to specify area types.
QualifiedVolumeTypeValue «CodeList»	QualifiedVolumeTypeValue is a code list used to specify volume types.
RelationTypeValue «CodeList»	RelationTypeValue is a code list used to classify city object relations.
TemporalRelationTypeValue «CodeList»	TemporalRelationTypeValue is a code list used to classify temporal city object relations.
TopologicRelationTypeValue «CodeList»	TopologicRelationTypeValue is a code list used to classify topological city object relations.

8.2.7. Additional Information

A detailed discussion of the CityGML Core can be found in the CityGML Best Practices document here.

8.3. Appearance

Requirements Class	
http://www.opengis.net/spec/CityGML/3.0/req/req-class-appearance	
Target type	Implementation Specification
Dependency	/req/req-class-core

The Appearance module provides the representation of surface data, i.e. observable properties for surface geometry objects in the form of textures and material. Appearances are not limited to visual data but represent arbitrary categories called themes such as infrared radiation, noise pollution, or earthquake-induced structural stress. 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).

Surface data that is constant across a surface is modelled as material based on the material definitions from the standards X3D and COLLADA. Surface data that depends on the exact location within the surface is modelled as texture. This can either be a parameterized texture, i.e. a texture that uses texture coordinates or a transformation matrix for parameterization, or a georeferenced texture, i.e. a texture that uses a planimetric projection. Each surface geometry object can have both, a material and a texture per theme and side. This allows for providing a constant approximation and a complex measurement of a surface's property simultaneously.

The UML diagram of the Appearance module is illustrated in Appearance UML Diagram. A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document here.

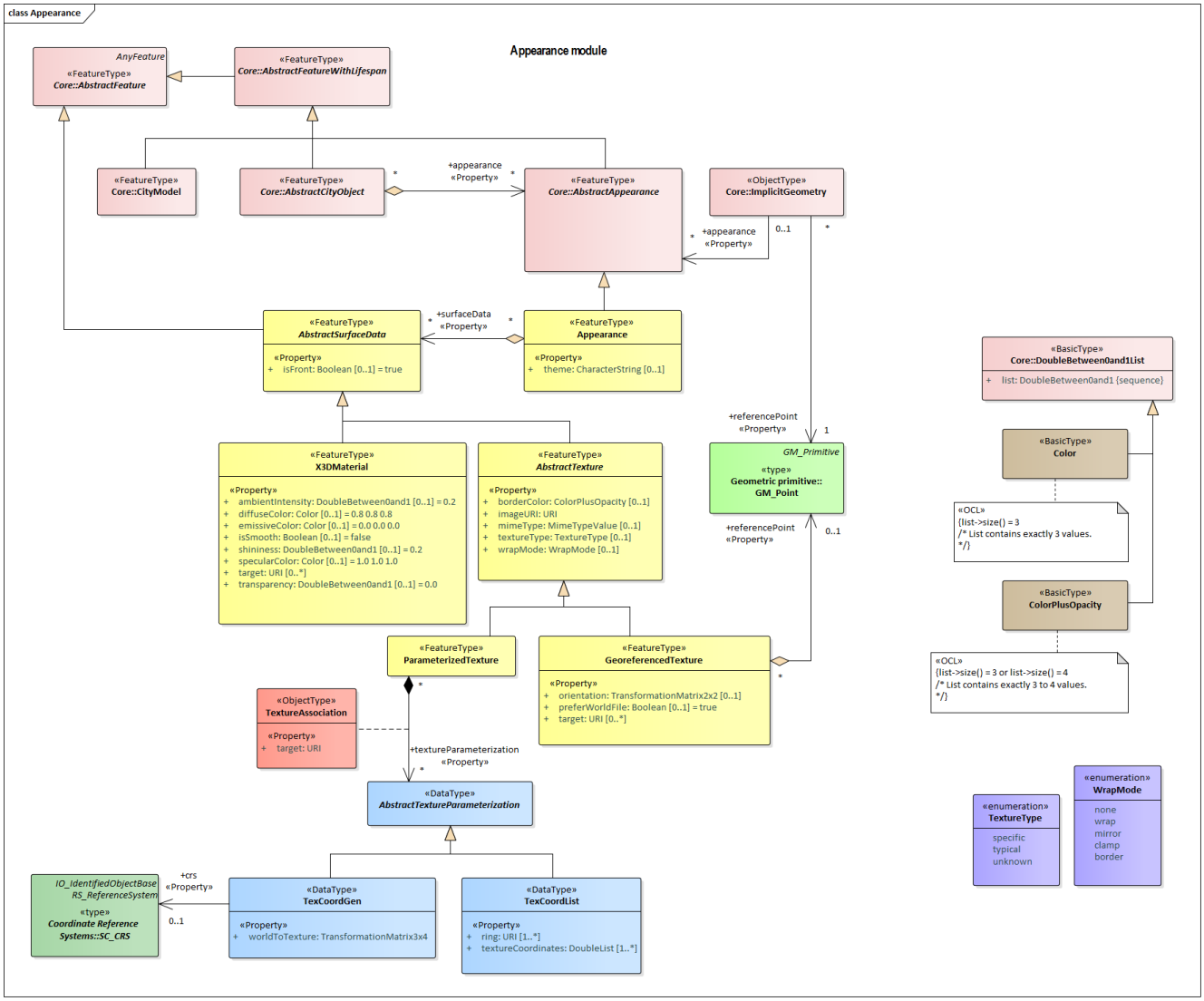

Figure 8. UML diagram of CityGML's appearance model.

Table 13. Classes used in Appearance

Class	Description
AbstractSurfaceData «FeatureType»	AbstractSurfaceData is the abstract superclass for different kinds of textures and material.
AbstractTexture «FeatureType»	AbstractTexture is the abstract superclass to represent the common attributes of the classes ParameterizedTexture and GeoreferencedTexture.

Appearance «FeatureType»	An Appearance is a collection of surface data, i.e. observable properties for surface geometry objects in the form of textures and material.
GeoreferencedTexture «FeatureType»	A GeoreferencedTexture is a texture that uses a planimetric projection. It contains an implicit parameterization that is either stored within the image file, an acompanying world file or specified using the orientation and referencePoint elements.
ParameterizedTexture «FeatureType»	A ParameterizedTexture is a texture that uses texture coordinates or a transformation matrix for parameterization.
X3DMaterial «FeatureType»	X3DMaterial defines properties for surface geometry objects based on the material definitions from the standards X3D and COLLADA.
TextureAssociation «ObjectType»	TextureAssociation denotes the relation of a texture to a surface geometry object.

Table 14. Data Types used in Appearance

Name	Description
Color «BasicType»	Color is a list of three double values between 0 and 1 defining an RGB color value.
ColorPlusOpacity «BasicType»	Color is a list of four double values between 0 and 1 defining an RGBA color value. Opacity value of 0 means transparent.
AbstractTextureParameterization «DataType»	AbstractTextureParameterization is the abstract superclass for different kinds of texture parameterizations.
TexCoordGen «DataType»	TexCoordGen defines texture parameterization using a transformation matrix.
TexCoordList «DataType»	TexCoordList defines texture parameterization using texture coordinates.

Table 15. Enumerated Classes used in Appearance

Name	Description
TextureType «Enumeration»	TextureType enumerates the different texture types.
WrapMode «Enumeration»	WrapMode enumerates the different fill modes for textures.

8.3.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.4. Bridge

Requirements Class

http://www.opengis.net/spec/CityGML/3.0/req/req-class-bridge	
Target type	Implementation Specification
Dependency	/req/req-class-core

The Bridge module provides the representation of thematic and spatial aspects of bridges. Bridges are movable or unmovable structures that span intervening natural or built elements. In this way, bridges allow the passage of pedestrians, animals, vehicles, and service(s) above obstacles or between two points at a height above ground. Bridges are represented in the UML model by the top-level feature type *Bridge*, which is the main class of the Bridge module. Bridges can physically or functionally be subdivided into bridge parts. In addition, bridges can be decomposed into structural elements, such as pylons, anchorages, cables, slabs, and beams.

The free space inside bridges is represented by rooms, which allows a virtual accessibility of bridges. Bridges can contain installations and furniture. Installations are permanent parts of a bridge that strongly affect the outer or inner appearance of the bridge and that cannot be moved. Examples are stairways, signals, railings, and lamps. Furniture, in contrast, represent moveable objects of a bridge, like signs, art works, and benches. Bridges can be bounded by different types of surfaces. In this way, the outer structure of bridges can be differentiated semantically into wall surfaces, roof surfaces, ground surfaces, outer floor surfaces, and outer ceiling surfaces, whereas the visible surface of rooms can be structured into interior wall surfaces, floor surfaces, and ceiling surfaces. Furthermore, the openings of bridges, i.e. windows and doors, can be represented including their corresponding surfaces.

The UML diagram of the Bridge module is depicted in Bridge UML Diagram. The Bridge module inherits concepts from the Construction module (cf. Section Construction). The Construction module defines objects that are common to all types of construction, such as the different surface types and the openings. A detailed discussion of Requirements Class Bridge can be found in the CityGML Best Practices document here.

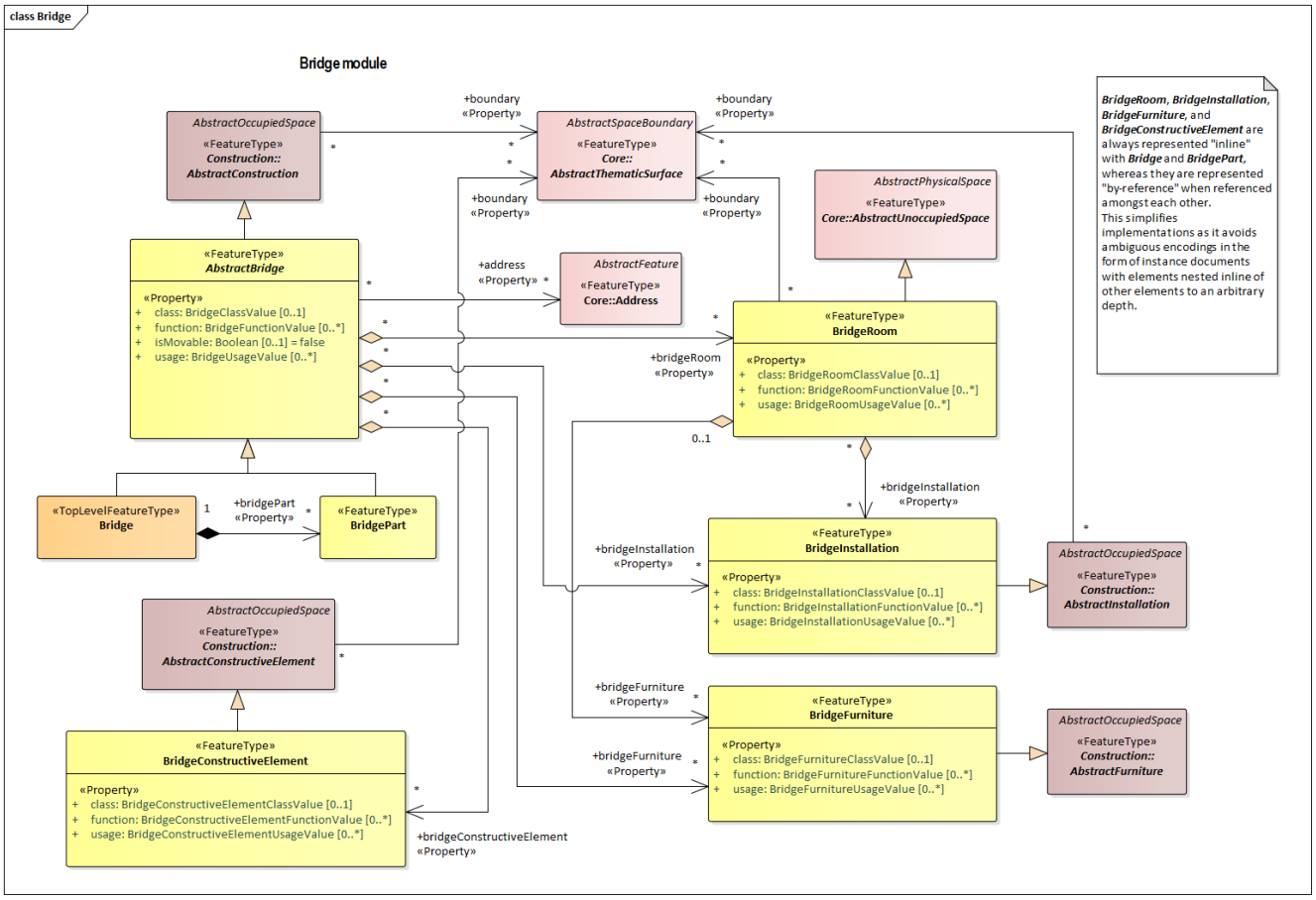

Figure 9. UML diagram of the Bridge Model.

Table 16. Classes used in Bridge

Class	Description
Bridge «TopLevelFeatureType»	A Bridge represents a structure that affords the passage of pedestrians, animals, vehicles, and service(s) above obstacles or between two points at a height above ground. [cf. ISO 6707-1]
AbstractBridge «FeatureType»	AbstractBridge is an abstract superclass representing the common attributes and associations of the classes Bridge and BridgePart.
BridgeConstructiveElement «FeatureType»	A BridgeConstructiveElement is an element of a bridge which is essential from a structural point of view. Examples are pylons, anchorages, slabs, beams.
BridgeFurniture «FeatureType»	A BridgeFurniture is an equipment for occupant use, usually not fixed to the bridge. [cf. ISO 6707-1]
BridgeInstallation «FeatureType»	A BridgeInstallation is a permanent part of a Bridge (inside and/or outside) which does not have the significance of a BridgePart. In contrast to BridgeConstructiveElements, a BridgeInstallation is not essential from a structural point of view. Examples are stairs, antennas or railways.
BridgePart «FeatureType»	A BridgePart is a physical or functional subdivision of a Bridge. It would be considered a Bridge, if it were not part of a collection of other BridgeParts.

BridgeRoom «FeatureType»	A BridgeRoom is a space within a Bridge or BridgePart intended for human occupancy (e.g. a place of work or recreation) and/or containment (storage) of animals or things. A BridgeRoom is bounded physically and/or virtually (e.g. by ClosureSurfaces or GenericSurfaces).

Table 17. CodeList Classes used in Bridge

Name	Description
BridgeClassValue «CodeList»	BridgeClassValue is a code list used to further classify a Bridge.
BridgeConstructiveElementClassValue «CodeList»	BridgeConstructiveElementClassValue is a code list used to further classify a BridgeConstructiveElement.
BridgeConstructiveElementFunctionValue «CodeList»	BridgeConstructiveElementFunctionValue is a code list that enumerates the different purposes of a BridgeConstructiveElement.
BridgeConstructiveElementUsageValue «CodeList»	BridgeConstructiveElementUsageValue is a code list that enumerates the different uses of a BridgeConstructiveElement.
BridgeFunctionValue «CodeList»	BridgeFunctionValue is a code list that enumerates the different purposes of a Bridge.
BridgeFurnitureClassValue «CodeList»	BridgeFurnitureClassValue is a code list used to further classify a BridgeFurniture.
BridgeFurnitureFunctionValue «CodeList»	BridgeFurnitureFunctionValue is a code list that enumerates the different purposes of a BridgeFurniture.
BridgeFurnitureUsageValue «CodeList»	BridgeFurnitureUsageValue is a code list that enumerates the different uses of a BridgeFurniture.
BridgeInstallationClassValue «CodeList»	BridgeInstallationClassValue is a code list used to further classify a BridgeInstallation.
BridgeInstallationFunctionValue «CodeList»	BridgeInstallationFunctionValue is a code list that enumerates the different purposes of a BridgeInstallation.
BridgeInstallationUsageValue «CodeList»	BridgeInstallationUsageValue is a code list that enumerates the different uses of a BridgeInstallation.
BridgeRoomClassValue «CodeList»	BridgeRoomClassValue is a code list used to further classify a BridgeRoom.

BridgeRoomFunctionValue «CodeList»	BridgeRoomFunctionValue is a code list that enumerates the different purposes of a BridgeRoom.
BridgeRoomUsageValue «CodeList»	BridgeRoomUsageValue is a code list that enumerates the different uses of a BridgeRoom.
BridgeUsageValue «CodeList»	BridgeUsageValue is a code list that enumerates the different uses of a Bridge.

8.4.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.5. Building

Requirements Class	
http://www.opengis.net/spec/CityGML/3.0/req/req-class-building	
Target type	Implementation Specification
Dependency	/req/req-class-core

The Building module provides the representation of thematic and spatial aspects of buildings. Buildings are free-standing, self-supporting constructions that are roofed and usually walled, and that can be entered by humans and are normally designed to stand permanently in one place. Buildings are intended for human occupancy (e.g. a place of work or recreation), habitation and/or shelter of humans, animals or things. Buildings are represented in the UML model by the top-level feature type *Building*, which is the main class of the Building module. Buildings can physically or functionally be subdivided into building parts and logically into storeys and building units (e.g. apartments). In addition, buildings can be decomposed into structural elements, such as walls, slabs, staircases, and beams.

The interior of buildings is represented by rooms. This 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. Buildings can contain installations and furniture. Installations are permanent parts of a building that strongly affect the outer or inner appearance of the building and that cannot be moved. Examples are balconies, chimneys, dormers or stairs. Furniture, in contrast, represent moveable objects inside a building, like tables and chairs. Buildings can be bounded by different types of surfaces. In this way, the outer façade of buildings can be differentiated semantically into wall surfaces, roof surfaces, ground surfaces, outer floor surfaces, and outer ceiling surfaces, whereas the visible surface of rooms can be structured into interior wall surfaces, floor surfaces, and ceiling surfaces. Furthermore, the openings of buildings, i.e. windows and doors, can be represented including their corresponding surfaces.

The UML diagram of the building module is depicted in Building UML Diagram. The Building module inherits concepts from the Construction module (cf. Section Construction). The

Construction module defines objects that are common to all types of construction, such as the different surface types and the openings. A detailed discussion of the Requirements Class Building can be found in the CityGML Best Practices document here.

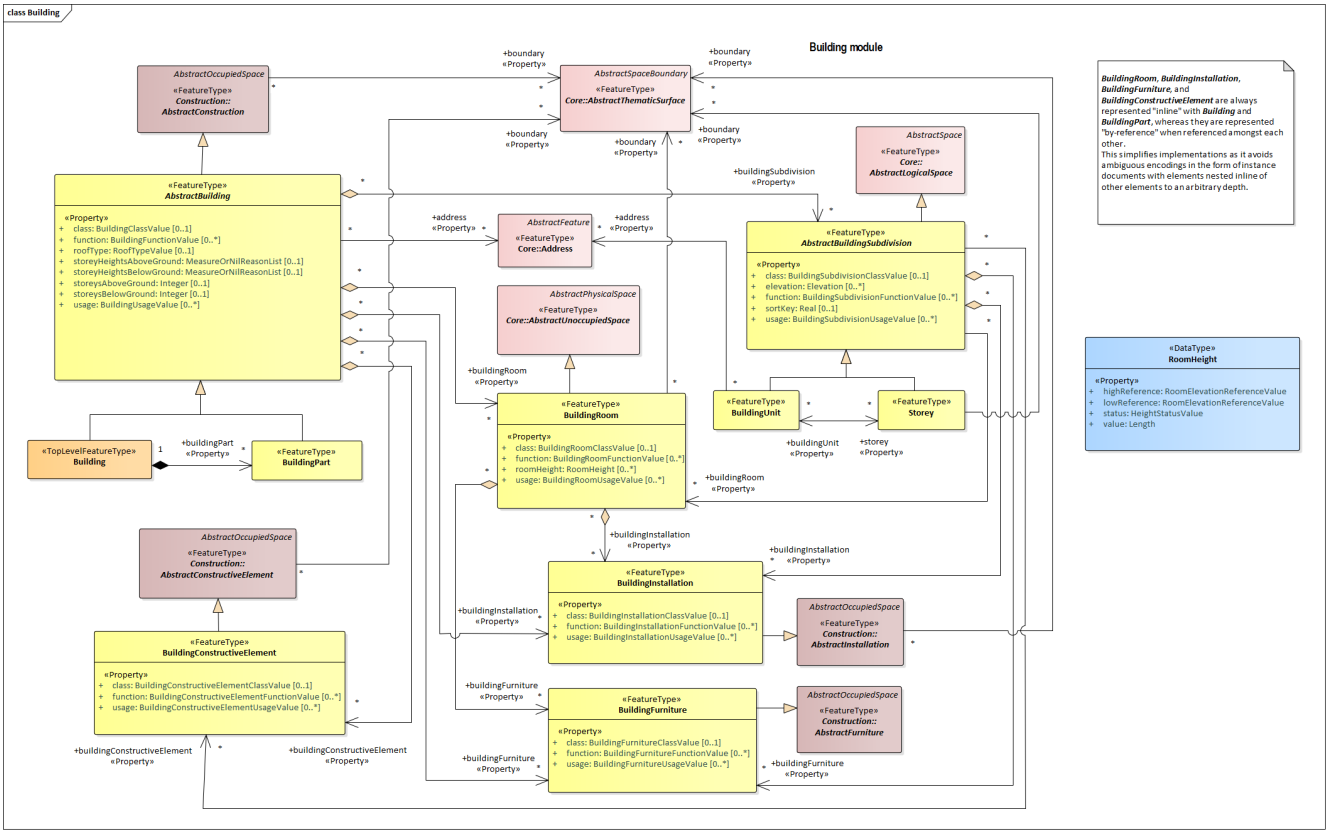

Figure 10. UML diagram of CityGML's building model.

Table 18. Classes used in Building

Class	Description
Building «TopLevelFeatureType»	A Building is a free-standing, self-supporting construction that is roofed, usually walled, and can be entered by humans and is normally designed to stand permanently in one place. It is intended for human occupancy (e.g. a place of work or recreation), habitation and/or shelter of humans, animals or things.
AbstractBuilding «FeatureType»	AbstractBuilding is an abstract superclass representing the common attributes and associations of the classes Building and BuildingPart.
AbstractBuildingSubdivision «FeatureType»	AbstractBuildingSubdivision is the abstract superclass for different kinds of logical building subdivisions.
BuildingConstructiveElement «FeatureType»	A BuildingConstructiveElement is an element of a Building which is essential from a structural point of view. Examples are walls, slabs, staircases, beams.
BuildingFurniture «FeatureType»	A BuildingFurniture is an equipment for occupant use, usually not fixed to the building. [cf. ISO 6707-1]

BuildingInstallation «FeatureType»	A BuildingInstallation is a permanent part of a Building (inside and/or outside) which has not the significance of a BuildingPart. Examples are stairs, antennas, balconies or small roofs.
BuildingPart «FeatureType»	A BuildingPart is a physical or functional subdivision of a Building. It would be considered a Building, if it were not part of a collection of other BuildingParts.
BuildingRoom «FeatureType»	A BuildingRoom is a space within a Building or BuildingPart intended for human occupancy (e.g. a place of work or recreation) and/or containment of animals or things. A BuildingRoom is bounded physically and/or virtually (e.g. by ClosureSurfaces or GenericSurfaces).
BuildingUnit «FeatureType»	A BuildingUnit is a logical subdivision of a Building. BuildingUnits are formed according to some homogeneous property like function, ownership, management, or accessability. They may be separately sold, rented out, inherited, managed, etc.
Storey «FeatureType»	A Storey is a horizontal section of a Building.

Table 19. Data Types used in Building

Name	Description
RoomHeight «DataType»	The RoomHeight represents a vertical distance (measured or estimated) between a low reference and a high reference. [cf. INSPIRE]

Table 20. CodeList Classes used in Building

Name	Description
BuildingClassValue «CodeList»	BuildingClassValue is a code list used to further classify a Building.
BuildingConstructiveElementClassValue «CodeList»	BuildingConstructiveElementClassValue is a code list used to further classify a BuildingConstructiveElement.
BuildingConstructiveElementFunctionValue «CodeList»	BuildingConstructiveElementFunctionValue is a code list that enumerates the different purposes of a BuildingConstructiveElement.
BuildingConstructiveElementUsageValue «CodeList»	BuildingConstructiveElementUsageValue is a code list that enumerates the different uses of a BuildingConstructiveElement.
BuildingFunctionValue «CodeList»	BuildingFunctionValue is a code list that enumerates the different purposes of a Building.
BuildingFurnitureClassValue «CodeList»	BuildingFurnitureClassValue is a code list used to further classify a BuildingFurniture.

BuildingFurnitureFunctionValue «CodeList»	BuildingFurnitureFunctionValue is a code list that enumerates the different purposes of a BuildingFurniture.
BuildingFurnitureUsageValue «CodeList»	BuildingFurnitureUsageValue is a code list that enumerates the different uses of a BuildingFurniture.
BuildingInstallationClassValue «CodeList»	BuildingInstallationClassValue is a code list used to further classify a BuildingInstallation.
BuildingInstallationFunctionValue «CodeList»	BuildingInstallationFunctionValue is a code list that enumerates the different purposes of a BuildingInstallation.
BuildingInstallationUsageValue «CodeList»	BuildingInstallationUsageValue is a code list that enumerates the different uses of a BuildingInstallation.
BuildingRoomClassValue «CodeList»	BuildingRoomClassValue is a code list used to further classify a BuildingRoom.
BuildingRoomFunctionValue «CodeList»	BuildingRoomFunctionValue is a code list that enumerates the different purposes of a BuildingRoom.
BuildingRoomUsageValue «CodeList»	BuildingRoomUsageValue is a code list that enumerates the different uses of a BuildingRoom.
BuildingSubdivisionClassValue «CodeList»	BuildingSubdivisionClassValue is a code list used to further classify a BuildingSubdivision.
BuildingSubdivisionFunctionValue «CodeList»	BuildingSubdivisionFunctionValue is a code list that enumerates the different purposes of a BuildingSubdivision.
BuildingSubdivisionUsageValue «CodeList»	BuildingSubdivisionUsageValue is a code list that enumerates the different uses of a BuildingSubdivision.
BuildingUsageValue «CodeList»	BuildingUsageValue is a code list that enumerates the different uses of a Building.
RoofTypeValue «CodeList»	RoofTypeValue is a code list that enumerates different roof types.
RoomElevationReferenceValue «CodeList»	RoomElevationReferenceValue is a code list that enumerates the different elevation reference levels used to measure room heights.

8.5.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.6. City Furniture

Requirements Class	
http://www.opengis.net/spec/CityGML/3.0/req/req-class-cityfurniture	
Target type	Implementation Specification
Dependency	/req/req-class-core

The CityFurniture module provides the representation of objects or pieces of equipment that are installed in the outdoor environment for various purposes, such as decoration, explanation or control. City furniture objects are relatively small, immovable objects and usually are of stereotypical form. Examples include road signs, traffic signals, bicycle racks, street lamps, fountains, flower buckets, advertising columns, and benches.

City furniture is represented in the UML model by the top-level feature type *CityFurniture*, which is also the only class of the CityFurniture module.

The UML diagram of the CityFurniture module is depicted in the City Furniture UML Diagram. A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document here.

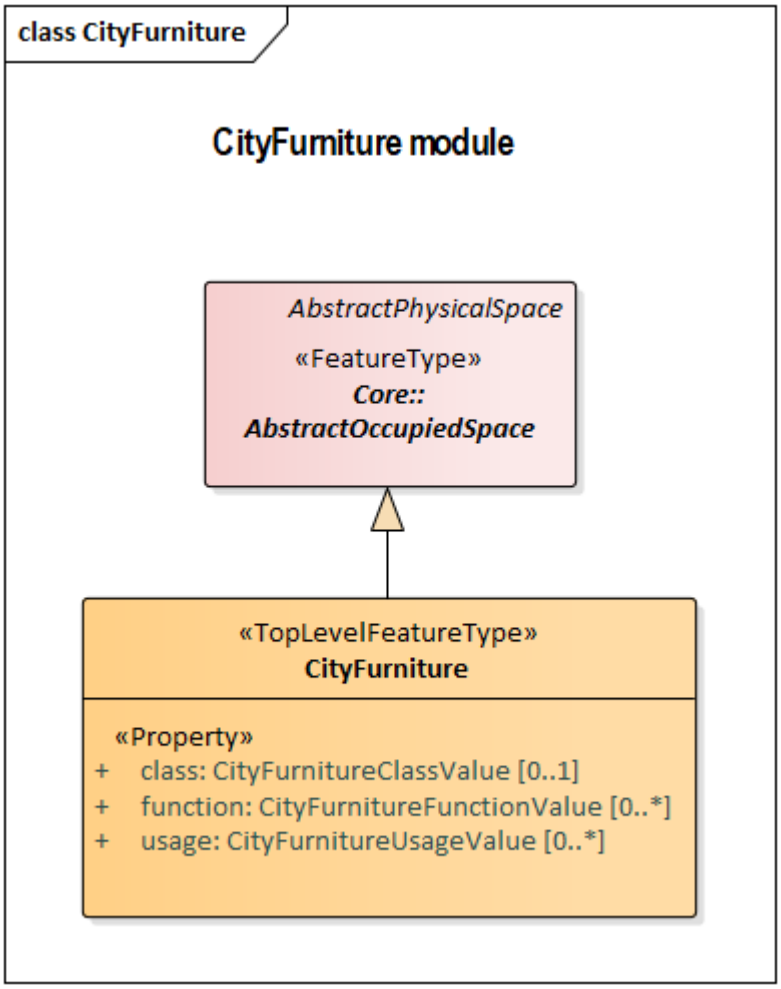

Figure 11. UML diagram of CityGML's City Furniture model.

Table 21. Classes used in CityFurniture

Class	Description
CityFurniture «TopLevelFeatureType»	CityFurniture is an object or piece of equipment installed in the outdoor environment for various purposes. Examples include street signs, traffic signals, street lamps, benches, fountains.
CityFurnitureClassValue «CodeList»	CityFurnitureClassValue is a code list used to further classify a CityFurniture.
CityFurnitureFunctionValue «CodeList»	CityFurnitureFunctionValue is a code list that enumerates the different purposes of a CityFurniture.
CityFurnitureUsageValue «CodeList»	CityFurnitureUsageValue is a code list that enumerates the different uses of a CityFurniture.

8.6.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.7. City Object Group

Requirements Class	
http://www.opengis.net/spec/CityGML/3.0/req/req-class-cityobjectgroup	
Target type	Implementation Specification
Dependency	/req/req-class-core

The CityObjectGroup module provides the application-specific aggregation of city objects 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 member of a group may be qualified by a role name, reflecting the role each city object plays in the context of the group. City object groups are represented in the UML model by the top-level feature type *CityObjectGroup*, which is the main class of the CityObjectGroup module.

City object groups can be linked to other city objects, the so-called parent objects, which allows for modelling a generic hierarchical grouping concept. In addition, as city object groups represent city objects themselves, a group may become a member of another group realizing recursive aggregation in this way.

The UML diagram of the CityObjectGroup module is depicted in City Object Group UML Diagram. A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document here.

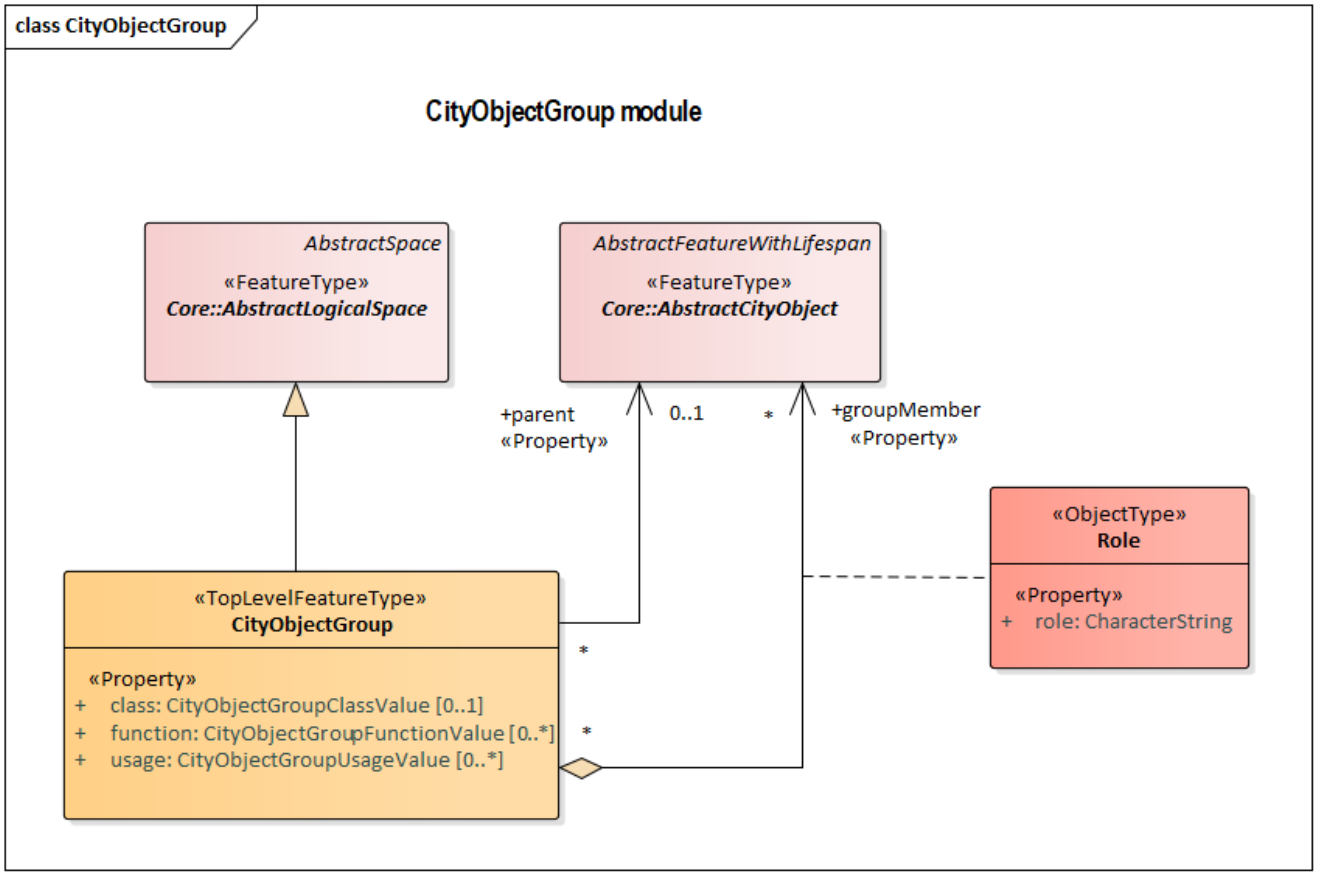

Figure 12. UML diagram of the City Object Group Model.

Table 22. Classes used in CityObjectGroup

Class	Description
CityObjectGroup «TopLevelFeatureType»	A CityObjectGroup represents an application-specific aggregation of city objects 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 member of a group may be qualified by a role name, reflecting the role each city object plays in the context of the group.
Role «ObjectType»	Role qualifies the function of a city object within the CityObjectGroup.
CityObjectGroupClassValue «CodeList»	CityObjectGroupClassValue is a code list used to further classify a CityObjectGroup.
CityObjectGroupFunctionValue «CodeList»	CityObjectGroupFunctionValue is a code list that enumerates the different purposes of a CityObjectGroup.
CityObjectGroupUsageValue «CodeList»	CityObjectGroupUsageValue is a code list that enumerates the different uses of a CityObjectGroup.

8.7.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.8. Construction

Requirements Class	
http://www.opengis.net/spec/CityGML/3.0/req/req-class-construction	
Target type	Implementation Specification
Dependency	/req/req-class-core

The Construction module defines concepts that are common to all kinds of constructions. Constructions are objects that are manufactured by humans from construction materials, are connected to earth and are intended to be permanent. The Construction module focuses on as-built representations of constructions and integrates all concepts that are similar over different types of constructions, in particular buildings, bridges, and tunnels. In addition, for representing man-made structures that are neither buildings, nor bridges, nor tunnels so-called other constructions (e.g. large chimneys or city walls) can be defined.

Furniture, installations, and constructive elements are further concepts that are defined in the Construction module. Installations are permanent parts of a construction that strongly affect the outer or inner appearance of the construction and that cannot be moved (e.g. balconies, chimneys, or stairs), whereas furniture represent moveable objects of a construction (e.g. tables and chairs). Constructive elements allow for decomposing a construction into volumetric components, such as

walls, beams, and slabs. Constructions and constructive elements can be bounded by different types of surfaces. In this way, the outer structure of constructions and constructive elements can be differentiated semantically into wall surfaces, roof surfaces, ground surfaces, outer floor surfaces, and outer ceiling surfaces, whereas the visible surface of interior spaces can be structured into interior wall surfaces, floor surfaces, and ceiling surfaces. Furthermore, the openings of constructions, i.e. windows and doors, can be represented as so-called filling elements including their corresponding filling surfaces.

The UML diagram of the Construction module is depicted in Construction UML Diagram. The Contruction module defines concepts that are inherited and, where necessary, are specialized by the modules Building, Bridge, and Tunnel (cf. sections Building, Bridge, and Tunnel). A detailed discussion of the Requirements Class Construction can be found in the CityGML Best Practices document here.

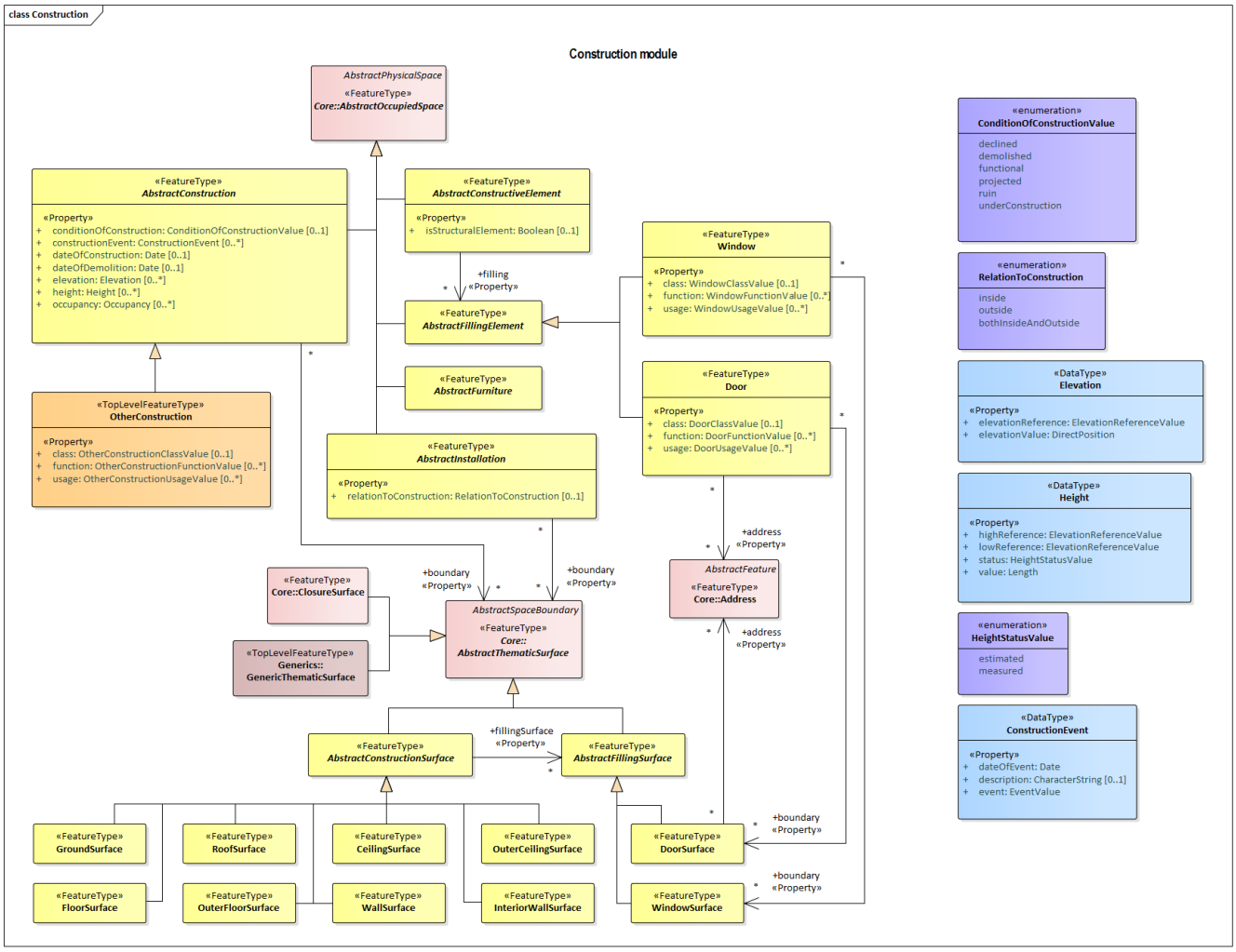

Figure 13. UML diagram of the Construction Model.

Table 23. Classes used in Construction

Class	Description
OtherConstruction «TopLevelFeatureType»	An OtherConstruction is a construction that is not covered by any of the other subclasses of AbstractConstruction.

AbstractConstruction «FeatureType»	AbstractConstruction is the abstract superclass for objects that are manufactured by humans from construction materials, are connected to earth and are intended to be permanent. A connection with the ground also exists when the construction rests by its own weight on the ground or is moveable limited on stationary rails or if the construction is intended to be used mainly stationary.
AbstractConstructionSurface «FeatureType»	AbstractConstructionSurface is the abstract superclass for different kinds of surfaces that bound a construction.
AbstractConstructiveElement «FeatureType»	AbstractConstructiveElement is the abstract superclass for the representation of volumetric elements of a construction. Examples are walls, beams, slabs.
AbstractFillingElement «FeatureType»	AbstractFillingElement is the abstract superclass for different kinds of elements that fill the openings of constructive elements.
AbstractFillingSurface «FeatureType»	AbstractFillingSurface is the abstract superclass for different kinds of surfaces that seal openings filled by filling elements.
AbstractFurniture «FeatureType»	AbstractFurniture is the abstract superclass for the representation of furniture objects of a construction.
AbstractInstallation «FeatureType»	AbstractInstallation is the abstract superclass for the representation of installation objects of a construction.
CeilingSurface «FeatureType»	A CeilingSurface is a surface that represents the interior ceiling of a construction. An example is the ceiling of a room.
Door «FeatureType»	A Door is a construction for closing an opening intended primarily for access or egress or both. [cf. ISO 6707-1]
DoorSurface «FeatureType»	A DoorSurface is either a boundary surface of a Door feature or a surface that seals an opening filled by a door.
FloorSurface «FeatureType»	A FloorSurface is surface that represents the interior floor of a construction. An example is the floor of a room.
GroundSurface «FeatureType»	A GroundSurface is a surface that represents the ground plate of a construction. The polygon defining the ground plate is congruent with the footprint of the construction.
InteriorWallSurface «FeatureType»	An InteriorWallSurface is a surface that is visible from inside a construction. An example is the wall of a room.
OuterCeilingSurface «FeatureType»	An OuterCeilingSurface is a surface that belongs to the outer building shell with the orientation pointing downwards. An example is the ceiling of a loggia.
OuterFloorSurface «FeatureType»	An OuterFloorSurface is a surface that belongs to the outer construction shell with the orientation pointing upwards. An example is the floor of a loggia.
RoofSurface «FeatureType»	A RoofSurface is a surface that delimits major roof parts of a construction.

WallSurface «FeatureType»	A WallSurface is a surface that is part of the building facade visible from the outside.
Window «FeatureType»	A Window is a construction for closing an opening in a wall or roof, primarily intended to admit light and/or provide ventilation. [cf. ISO 6707-1]

Table 24. Data Types used in Construction

Name	Description
ConstructionEvent «DataType»	A ConstructionEvent is a data type used to describe a specific event that is associated with a construction. Examples are the issuing of a building permit or the renovation of a building.
Elevation «DataType»	Elevation is a data type that includes the elevation value itself and information on how this elevation was measured. [cf. INSPIRE]
Height «DataType»	Height represents a vertical distance (measured or estimated) between a low reference and a high reference. [cf. INSPIRE]

Table 25. Enumerated Classes used in Construction

Name	Description
HeightStatusValue «Enumeration»	HeightStatusValue enumerates the different methods used to capture a height. [cf. INSPIRE]
RelationToConstruction «Enumeration»	RelationToConstruction is an enumeration used to describe whether an installation is positioned inside and/or outside of a construction.

Table 26. CodeList Classes used in Construction

Name	Description
DoorClassValue «CodeList»	DoorClassValue is a code list used to further classify a Door.
DoorFunctionValue «CodeList»	DoorFunctionValue is a code list that enumerates the different purposes of a Door.
DoorUsageValue «CodeList»	DoorUsageValue is a code list that enumerates the different uses of a Door.
ElevationReferenceValue «CodeList»	ElevationReferenceValue is a code list that enumerates the different elevation reference levels used to measure construction heights.
EventValue «CodeList»	EventValue is a code list that enumerates the different events of a construction.
OtherConstructionClassValue «CodeList»	OtherConstructionClassValue is a code list used to further classify an OtherConstruction.

OtherConstructionFunctionValue «CodeList»	OtherConstructionFunctionValue is a code list that enumerates the different purposes of an OtherConstruction.
OtherConstructionUsageValue «CodeList»	OtherConstructionUsageValue is a code list that enumerates the different uses of an OtherConstruction.
WindowClassValue «CodeList»	WindowClassValue is a code list used to further classify a Window.
WindowFunctionValue «CodeList»	WindowFunctionValue is a code list that enumerates the different purposes of a Window.
WindowSurface «FeatureType»	A WindowSurface is either a boundary surface of a Window feature or a surface that seals an opening filled by a window.
WindowUsageValue «CodeList»	WindowUsageValue is a code list that enumerates the different uses of a Window.
ConditionOfConstructionValue	ConditionOfConstructionValue enumerates different conditions of a construction. [cf. INSPIRE]

8.8.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.9. Dynamizer

Requirements Class	
http://www.opengis.net/spec/CityGML/3.0/req/req-class-dynamizer	
Target type	Implementation Specification
Dependency	/req/req-class-core

The Dynamizer module provides the concepts that allow for representing time-varying data for city object properties as well as for integrating sensors with 3D city models. Dynamizers are objects that inject timeseries data for an individual attribute of the city object in which it is included. The timeseries data overrides the static value of the referenced city object attribute in order to represent dynamic, i.e. time-dependent, variations of its value.

The dynamic values may be given by retrieving observation results directly from external sensor/IoT services using a sensor connection (e.g. OGC SensorThings API, Sensor Observation Service, or other sensor data platforms including MQTT). Alternatively, the dynamic values may be provided as atomic timeseries that represent time-varying data of a specific data type for a single contiguous time interval. The data can be provided in external tabulated files, such as CSV or Excel sheets, in external files that format timeseries data according to the OGC TimeseriesML or OGC Observations & Measurements standards, or inline as embedded time-value-pairs. Furthermore, timeseries data can also be aggregated to form composite timeseries with non-overlapping time intervals.

By using the Dynamizer module, fast changes over a short or longer time period with respect to cities and city models can be represented. This includes variations of spatial properties, i.e. change of a feature's geometry, both in respect to shape and to location (e.g. moving objects), variations of thematic attributes, i.e. changes of physical quantities like energy demands, temperatures, solar irradiation, traffic density, pollution concentration, or overpressure on building walls, and variations with respect to sensor or real-time data resulting from simulations or measurements.

The UML diagram of the Dynamizer module is depicted in Dynamizer UML Diagram. A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document here.

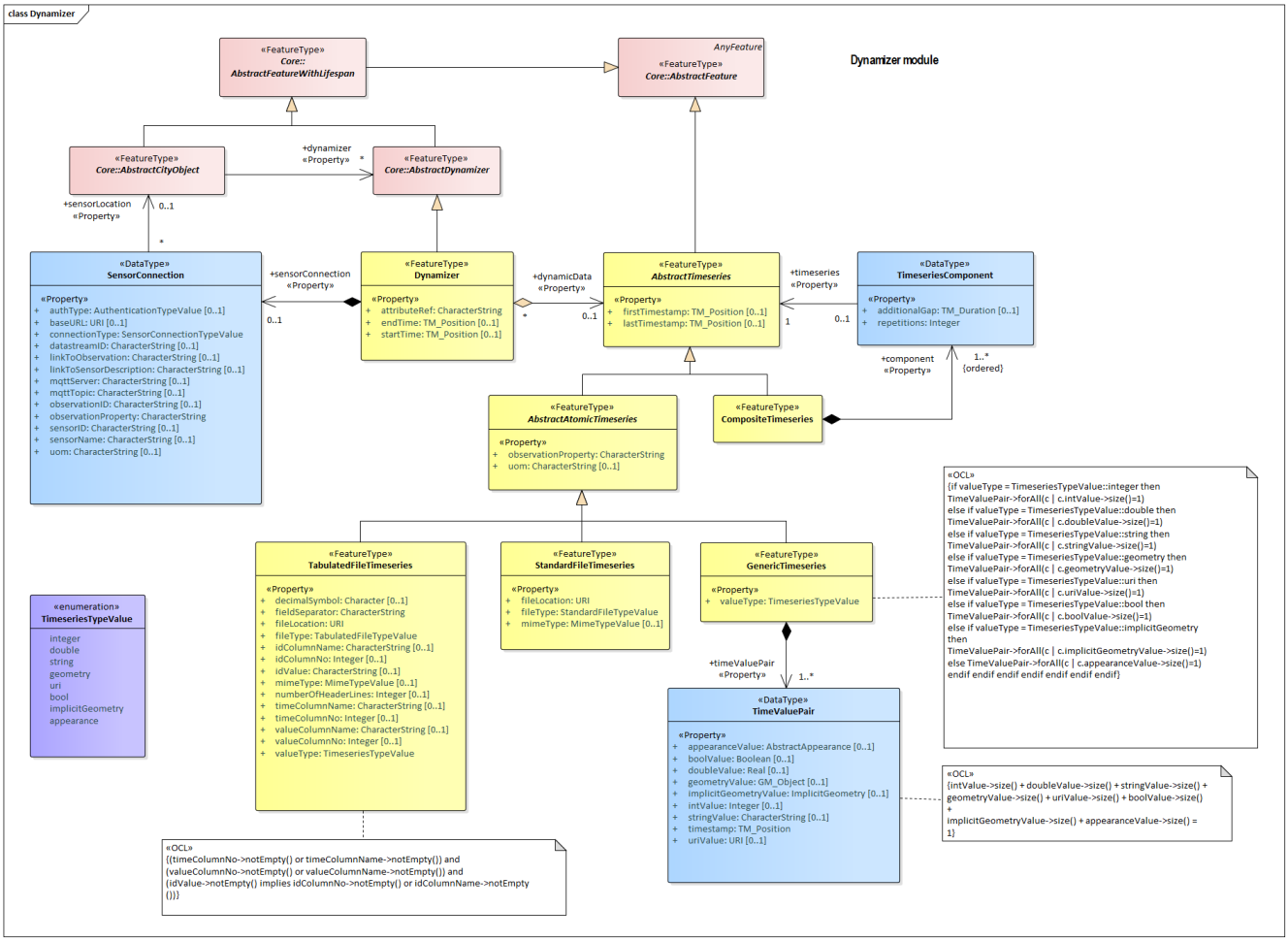

Figure 14. UML diagram of the Dynamizer Model.

Table 27. Classes used in Dynamizer

Class	Description
AbstractAtomicTimeseries «FeatureType»	AbstractAtomicTimeseries represents the attributes and relationships that are common to all kinds of atomic timeseries (GenericTimeseries, TabulatedFileTimeseries, StandardFileTimeseries). An atomic timeseries represents time-varying data of a specific data type for a single contiguous time interval.
AbstractTimeseries «FeatureType»	AbstractTimeseries is the abstract superclass representing any type of timeseries data.

CompositeTimeseries «FeatureType»	A CompositeTimeseries is a (possibly recursive) aggregation of atomic and composite timeseries. The components of a composite timeseries must have non-overlapping time intervals.
Dynamizer «FeatureType»	A Dynamizer is an object that injects timeseries data for an individual attribute of the city object in which it is included. The timeseries data overrides the static value of the referenced city object attribute in order to represent dynamic, i.e. time-dependent, variations of its value.
GenericTimeseries «FeatureType»	A GenericTimeseries represents time-varying data in the form of embedded time-value-pairs of a specific data type for a single contiguous time interval.
StandardFileTimeseries «FeatureType»	A StandardFileTimeseries represents time-varying data for a single contiguous time interval. The data is provided in an external file referenced in the StandardFileTimeseries. The data within the external file shall be encoded according to a dedicated format for the representation of timeseries data, for example, the OGC TimeseriesML or OGC Observations & Measurements standard. The data type of the data has to be specified within the external file.
TabulatedFileTimeseries «FeatureType»	A TabulatedFileTimeseries represents time-varying data of a specific data type for a single contiguous time interval. The data is provided in an external file referenced in the TabulatedFileTimeseries. The file shall contain table structured data using an appropriate file format like comma separated values (CSV), Microsoft Excel (XLSX) or Google Spreadsheet. The timestamps and the values are given in specific columns of the table. Each row represents a single time-value-pair. A subset of rows can be selected using the idColumn and idValue attributes.

Table 28. Data Types used in Dynamizer

Name	Description
SensorConnection «DataType»	A SensorConnection provides all details that are required to retrieve a specific datastream from an external sensor web service. It comprises the service type (e.g. OGC SensorThings API, OGC Sensor Observation Services, MQTT, proprietary platforms), the URL of the sensor service, the identifier for the sensor or thing and its observed property as well as information about the required authentication method.
TimeseriesComponent «DataType»	TimeseriesComponent represents an element of a CompositeTimeseries.
TimeValuePair «DataType»	A TimeValuePair represents a value that is valid for a given timepoint. For each TimeValuePair only one of the value properties can be used mutually exclusive. Which value property has to be provided depends on the selected value type in the GenericTimeSeries feature, in which the TimeValuePair is included.

Table 29. Enumerated Classes used in Dynamizer

Name	Description

TimeseriesTypeValue «Enumeration»	TimeseriesTypeValue enumerates the possible value types for GenericTimeseries and TimeValuePair.

Table 30. CodeList Classes used in Dynamizer

Name	Description
AuthenticationTypeValue «CodeList»	AuthenticationTypeValue is a code list used to specify the authentication method to be used to access the referenced sensor service. Each value shall provide enough information such that a software application could determine the required access credentials.
SensorConnectionTypeValue «CodeList»	SensorConnectionTypeValue is a code list used to specify the type of the referenced sensor service. Each value shall provide enough information such that a software application would be able to identify the API type and version.
StandardFileTypeValue «CodeList»	StandardFileTypeValue is a code list used to specify the type of the referenced external timeseries data file. Each value shall provide information about the standard and version.
TabulatedFileTypeValue «CodeList»	TabulatedFileTypeValue is a code list used to specify the data format of the referenced external tabulated data file.

8.9.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.10. Generics

Requirements Class	
http://www.opengis.net/spec/CityGML/3.0/req/req-class-generics	
Target type	Implementation Specification
Dependency	/req/req-class-core

The Generics module provides the representation of generic city objects, i.e. city objects that are not covered by any explicitly modelled thematic class within CityGML, and of generic attributes, i.e. attributes that are not explicitly represented in CityGML. In order to avoid problems concerning semantic interoperability, generic city objects and generic attributes shall only be used if appropriate thematic classes and attributes are not provided by any other CityGML module.

In accordance with the CityGML Space concept defined in the Core module (cf. Section Core) generic city objects can be represented as generic logical spaces, generic occupied spaces, generic unoccupied spaces, and generic thematic surfaces. In this way, spaces and surfaces can be defined that are not represented by any explicitly modelled class within CityGML that is a subclass of the classes AbstractLogicalSpace, AbstractOccupiedSpace, AbstractUnoccupiedSpace or AbstractThematicSurface, respectively. Generic city objects are represented in the UML model by

the top-level feature types *GenericLogicalSpace*, *GenericOccupiedSpace*, *GenericUnoccupiedSpace* and *GenericThematicSurface*.

Generic attributes are defined as name-value pairs and are always associated with a city object. Generic attributes can be of type String, Integer, Double, Date, URI, and Measure. In addition, generic attributes can be grouped under a common name as generic attribute sets.

The UML diagram of the Generics module is depicted in Generics UML Diagram. A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document here.

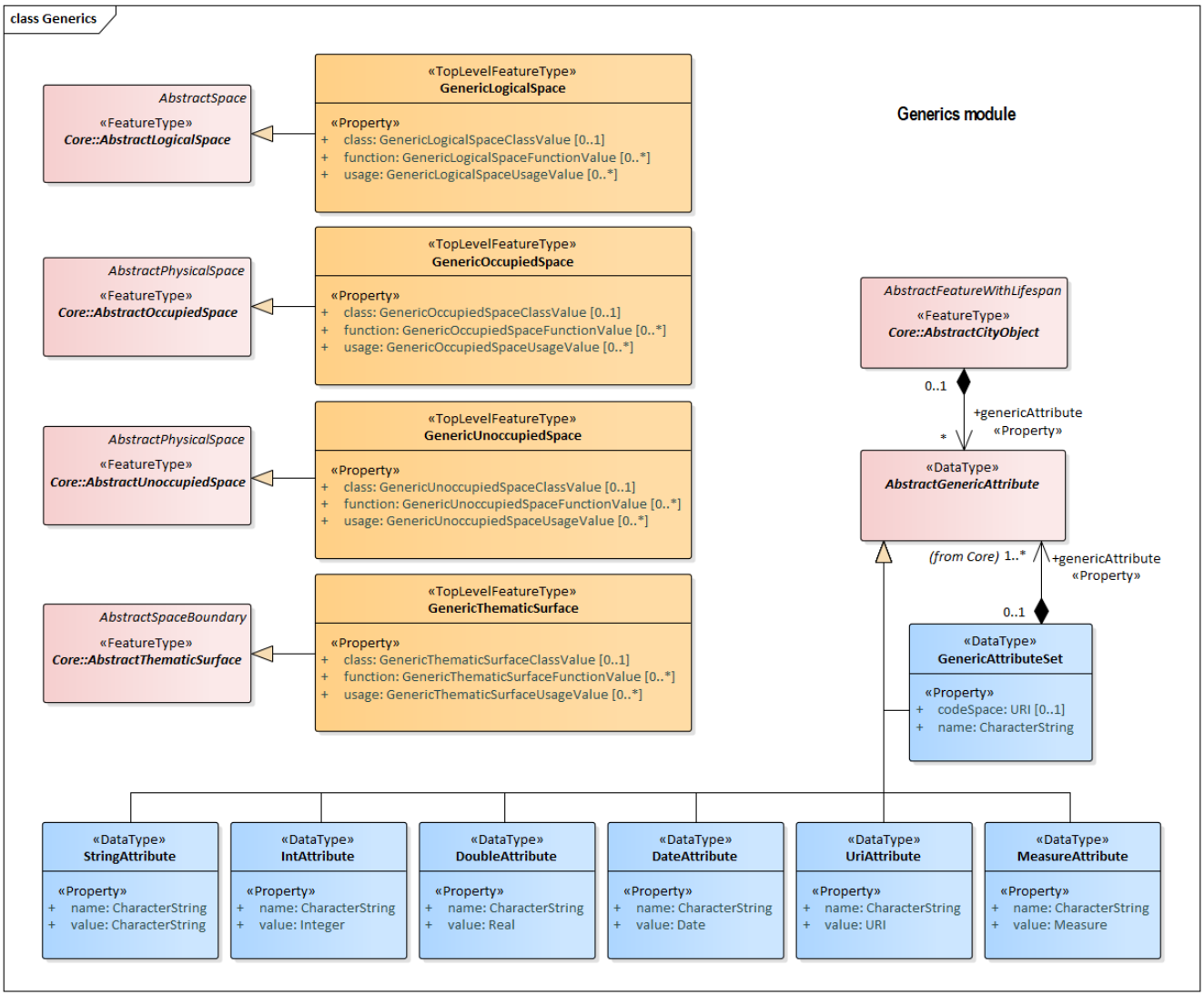

Figure 15. UML diagram of the Generics Model.

Table 31. Classes used in Generics

Class	Description
GenericLogicalSpace «TopLevelFeatureType»	A GenericLogicalSpace is a space that is not represented by any explicitly modelled AbstractLogicalSpace subclass within CityGML.
GenericOccupiedSpace «TopLevelFeatureType»	A GenericOccupiedSpace is a space that is not represented by any explicitly modelled AbstractOccupiedSpace subclass within CityGML.

GenericThematicSurface «TopLevelFeatureType»	A GenericThematicSurface is a surface that is not represented by any explicitly modelled AbstractThematicSurface subclass within CityGML.
GenericUnoccupiedSpace «TopLevelFeatureType»	A GenericUnoccupiedSpace is a space that is not represented by any explicitly modelled AbstractUnoccupiedSpace subclass within CityGML.

Table 32. Data Types used in Generics

Name	Description
DateAttribute «DataType»	DateAttribute is a data type used to define generic attributes of type "Date".
DoubleAttribute «DataType»	DoubleAttribute is a data type used to define generic attributes of type "Double".
GenericAttributeSet «DataType»	A GenericAttributeSet is a named collection of generic attributes.
IntAttribute «DataType»	IntAttribute is a data type used to define generic attributes of type "Integer".
MeasureAttribute «DataType»	MeasureAttribute is a data type used to define generic attributes of type "Measure".
StringAttribute «DataType»	StringAttribute is a data type used to define generic attributes of type "String".
UriAttribute «DataType»	UriAttribute is a data type used to define generic attributes of type "URI".

Table 33. CodeList Classes used in Generics

Name	Description
GenericLogicalSpaceClassValue «CodeList»	GenericLogicalSpaceClassValue is a code list used to further classify a GenericLogicalSpace.
GenericLogicalSpaceFunctionValue «CodeList»	GenericLogicalSpaceFunctionValue is a code list that enumerates the different purposes of a GenericLogicalSpace.
GenericLogicalSpaceUsageValue «CodeList»	GenericLogicalSpaceUsageValue is a code list that enumerates the different uses of a GenericLogicalSpace.
GenericOccupiedSpaceClassValue «CodeList»	GenericOccupiedSpaceClassValue is a code list used to further classify a GenericOccupiedSpace.
GenericOccupiedSpaceFunctionValue «CodeList»	GenericOccupiedSpaceFunctionValue is a code list that enumerates the different purposes of a GenericOccupiedSpace.

GenericOccupiedSpaceUsageValue «CodeList»	GenericOccupiedSpaceUsageValue is a code list that enumerates the different uses of a GenericOccupiedSpace.
GenericThematicSurfaceClassValue «CodeList»	GenericThematicSurfaceClassValue is a code list used to further classify a GenericThematicSurface.
GenericThematicSurfaceFunctionValue «CodeList»	GenericThematicSurfaceFunctionValue is a code list that enumerates the different purposes of a GenericThematicSurface.
GenericThematicSurfaceUsageValue «CodeList»	GenericThematicSurfaceUsageValue is a code list that enumerates the different uses of a GenericThematicSurface.
GenericUnoccupiedSpaceClassValue «CodeList»	GenericUnoccupiedSpaceClassValue is a code list used to further classify a GenericUnoccupiedSpace.
GenericUnoccupiedSpaceFunctionValue «CodeList»	GenericUnoccupiedSpaceFunctionValue is a code list that enumerates the different purposes of a GenericUnoccupiedSpace.
GenericUnoccupiedSpaceUsageValue «CodeList»	GenericUnoccupiedSpaceUsageValue is a code list that enumerates the different uses of a GenericUnoccupiedSpace.

8.10.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.11. Land Use

Requirements Class	
http://www.opengis.net/spec/CityGML/3.0/req/req-class-landuse	
Target type	Implementation Specification
Dependency	/req/req-class-core

The LandUse module defines objects that can be used to describe areas of the earth's surface dedicated to a specific land use or having a specific land cover with or without vegetation, such as sand, rock, mud flats, forest, grasslands, or wetlands (i.e. the physical appearance). Land use and land cover are different concepts; the first describes human activities on the earth's surface, the second describes its physical and biological cover. However, the two concepts are interlinked and often mixed in practice. Land use objects in CityGML support both concepts: They can be employed to represent parcels, spatial planning objects, recreational objects and objects describing the physical characteristics of an area in 3D. Land use objects are represented in the UML model by the top-level feature type *LandUse*, which is also the only class of the LandUse module.

The UML diagram of the LandUse module is depicted in Land Use UML Diagram. A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document here.

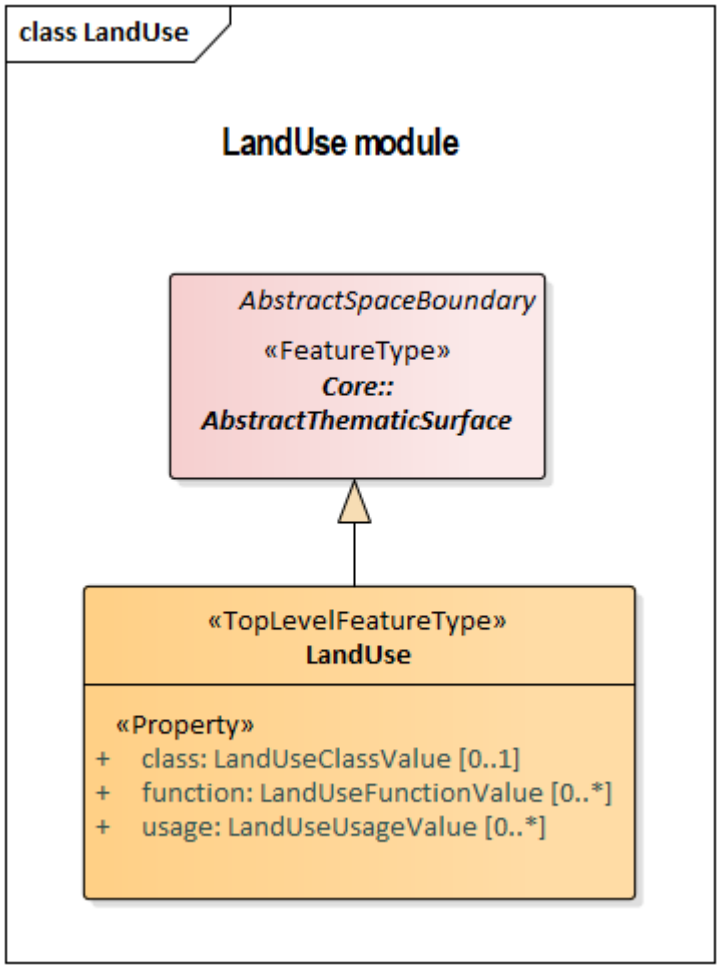

Figure 16. UML diagram of the Land Use Model.

Table 34. Classes used in LandUse

Class	Description
LandUse «TopLevelFeatureType»	A LandUse object is an area of the earth's surface dedicated to a specific land use or having a specific land cover with or without vegetation, such as sand, rock, mud flats, forest, or grasslands.
LandUseClassValue «CodeList»	LandUseClassValue is a code list used to further classify a LandUse.
LandUseFunctionValue «CodeList»	LandUseFunctionValue is a code list that enumerates the different purposes of a LandUse.
LandUseUsageValue «CodeList»	LandUseUsageValue is a code list that enumerates the different uses of a LandUse.

8.11.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.12. Point Cloud

Requirements Class	
http://www.opengis.net/spec/CityGML/3.0/req/req-class-pointcloud	
Target type	Implementation Specification
Dependency	/req/req-class-core

The PointCloud module offers the possibility to provide the geometry of physical spaces and of thematic surfaces by 3D point clouds. In this way, the building hull, a room within a building or a single wall surface can be spatially represented by a point cloud only. The same applies to all other thematic feature types including transportation objects, vegetation, city furniture, etc. Point clouds can either be provided inline within a CityGML file or as reference to external point cloud files of common file types such as LAS or LAZ. Point clouds are represented in the UML model by the feature type *PointCloud*, which is also the only class of the PointCloud module.

The UML diagram of the PointCloud module is depicted in Point Cloud UML Diagram. A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document here.

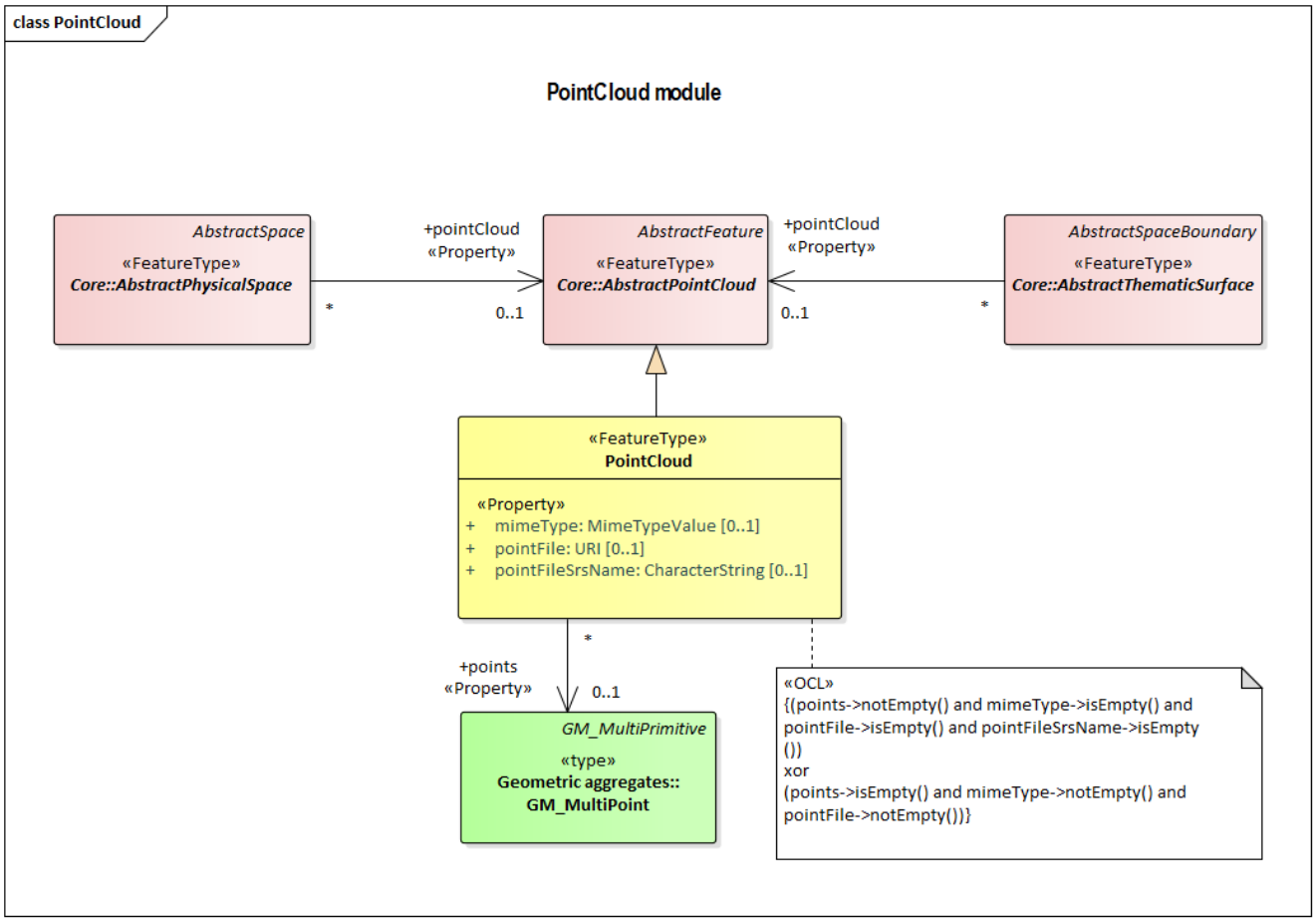

Figure 17. UML diagram of the Point Cloud Model.

Table 35. Classes used in PointCloud

Class	Description

PointCloud «FeatureType»	A PointCloud is an unordered collection of points that is a sampling of the geometry of a space or space boundary.

8.12.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.13. Digital Terrain Model

Requirements Class	
http://www.opengis.net/spec/CityGML/3.1/req/req-class-relief	
Target type	Implementation Specification
Dependency	/req/req-class-core

The Relief module provides the representation of terrain which is an essential part of city models. In CityGML, the terrain is modelled by relief features. They are represented in the UML model by the top-level feature type *ReliefFeature*, which is the main class of the Relief module. The relief features, in turn, are collections of relief components that describe the Earth's surface, a.k.a. the Digital Terrain Model. The relief components can have different terrain representations which can coexist. Each relief component may be specified as a regular raster or grid, as a TIN (Triangulated Irregular Network), by break lines, or by mass points. In addition, the validity of the relief components may be restricted to certain areas.

The UML diagram of the Relief module is depicted in Relief UML Diagram. A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document here.

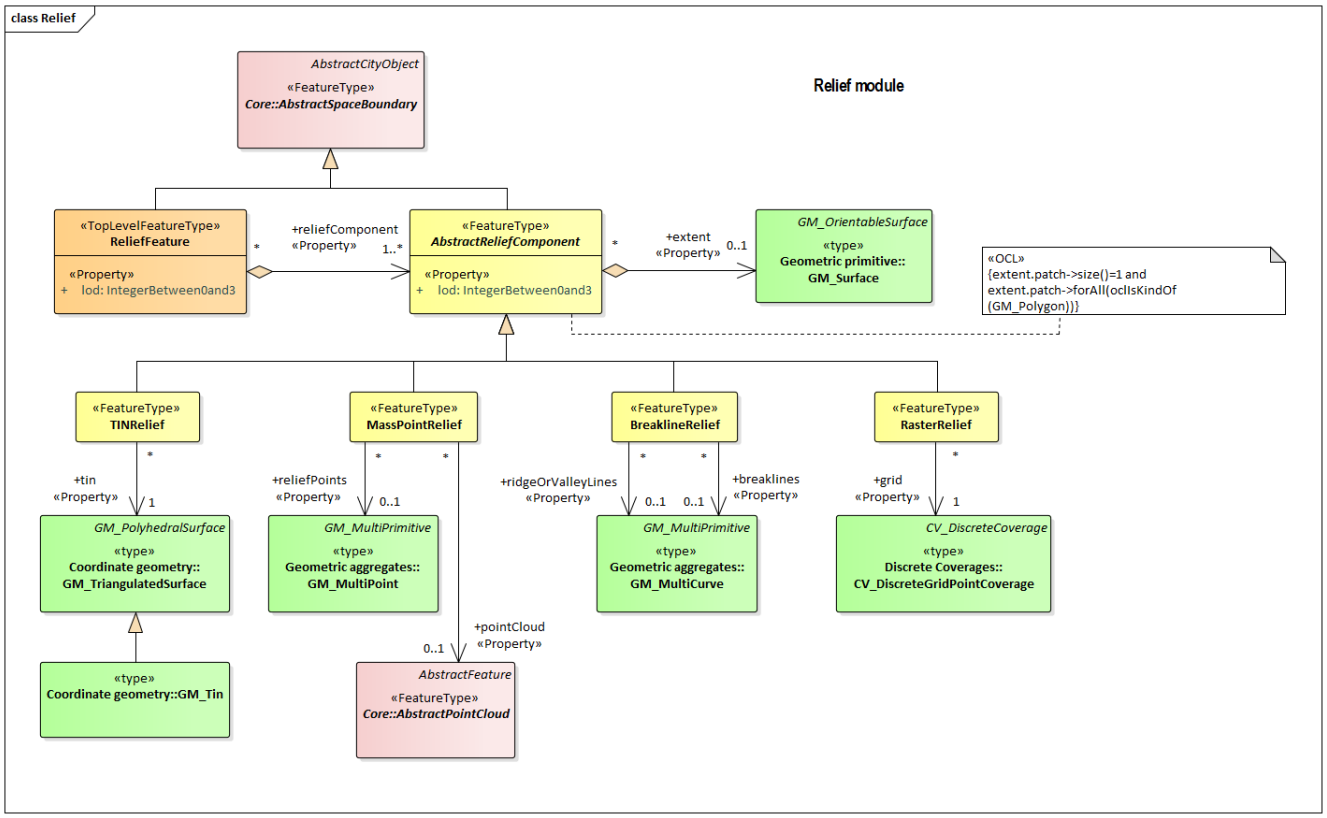

Figure 18. UML diagram of Relief module.

Table 36. Classes used in Relief

Class	Description
ReliefFeature «TopLevelFeatureType»	A ReliefFeature is a collection of terrain components representing the Earth's surface, a.k.a. the Digital Terrain Model.
AbstractReliefComponent «FeatureType»	An AbstractReliefComponent represents an element of the terrain surface - either a TIN, a Grid, mass points or break lines.
BreaklineRelief «FeatureType»	A BreaklineRelief represents a terrain component with 3D lines. These lines denote break lines or ridge/valley lines.
MassPointRelief «FeatureType»	A MassPointRelief represents a terrain component as a collection of 3D points.
RasterRelief «FeatureType»	A RasterRelief represents a terrain component as a regular raster or grid.
TINRelief «FeatureType»	A TINRelief represents a terrain component as a triangulated irregular network.

8.13.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.14. Transportation

Requirements Class	
http://www.opengis.net/spec/CityGML/3.0/req/req-class-transportation	
Target type	Implementation Specification
Dependency	/req/req-class-core

The Transportation module defines central elements of the traffic infrastructure. This includes the transportation objects road, track, and square for the movement of vehicles, bicycles, and pedestrians, the transportation object railway for the movement of wheeled vehicles on rails, as well as the transportation object waterway for the movement of vessels upon or within water bodies. The transportation objects are represented in the UML model by the top-level feature types *Road*, *Track*, *Square*, *Railway*, and *Waterway*, which are the main classes of the Transportation module. Transportation objects can be subdivided into sections, which can be regular road, track or railway legs, into intersection areas, and into roundabouts.

For each transportation object, traffic spaces and auxiliary traffic spaces can be provided, which are bounded at the bottom by traffic areas and auxiliary traffic areas, respectively. Traffic areas are elements that are important in terms of traffic usage, such as driving lanes, sidewalks, and cycle lanes, whereas auxiliary traffic areas describe further elements, such as kerbstones, middle lanes, and green areas. The corresponding spaces define the free space above the areas. In addition, each traffic space can have an optional clearance space. The transportation objects can be represented in different levels of granularity, either as a single area, split up into individual lanes or even decomposed into individual (carriage)ways. Furthermore, holes in the surfaces of roads, tracks or squares, such as road damages, manholes or drains, can be represented including their corresponding boundary surfaces. In addition, markings for the structuring or restriction of traffic can be added to the transportation areas. Examples are road markings and markings related to railway or waterway traffic.

The UML diagram of the Transportation module is depicted in Transportation UML Diagram. A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document here.

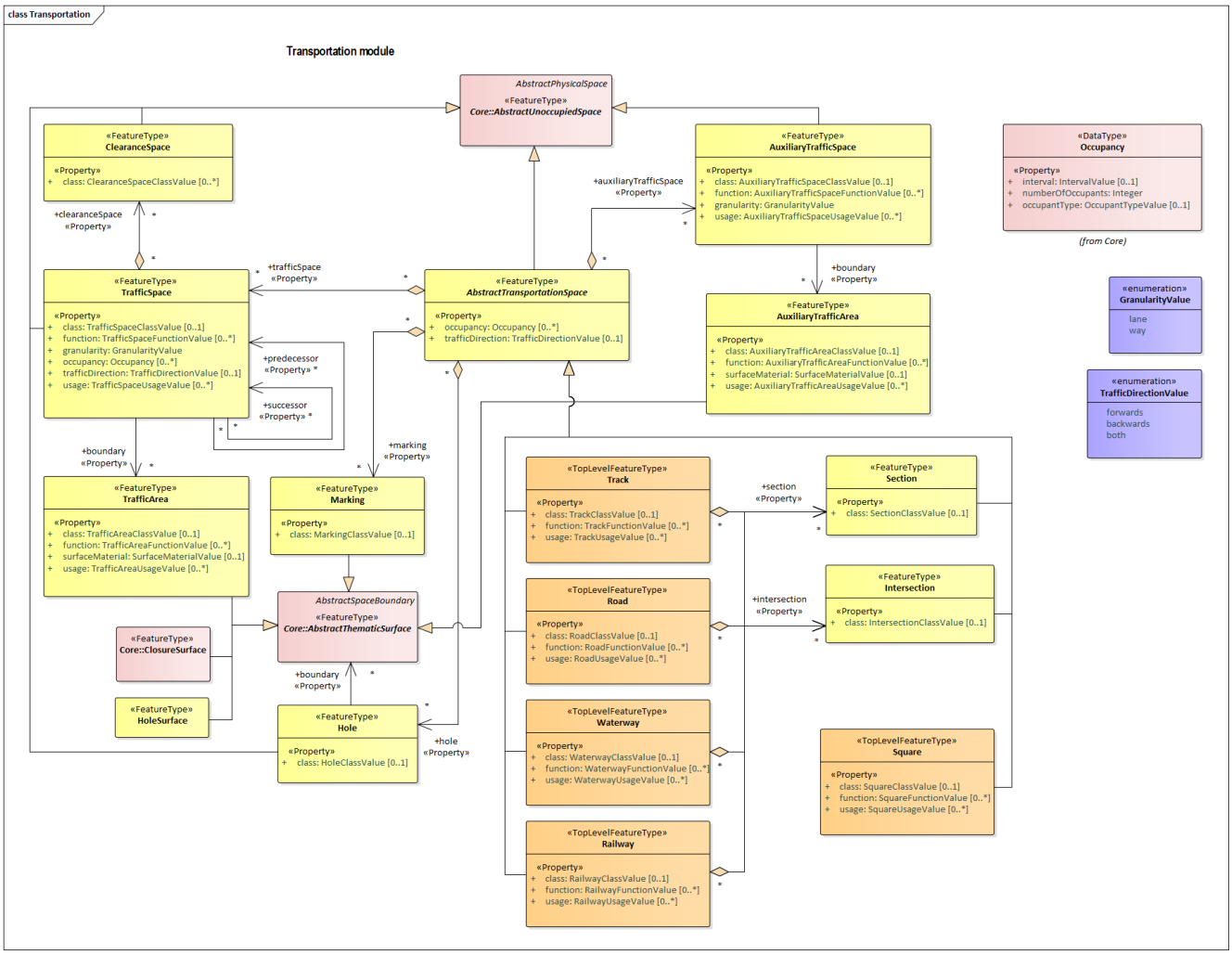

Figure 19. UML diagram of the Transportation Model.

Table 37. Classes used in Transportation

Class	Description
Railway «TopLevelFeatureType»	A Railway is a transportation space used by wheeled vehicles on rails.
Road «TopLevelFeatureType»	A Road is a transportation space used by vehicles, bicycles and/or pedestrians.
Square «TopLevelFeatureType»	A Square is a transportation space for unrestricted movement for vehicles, bicycles and/or pedestrians. This includes plazas as well as large sealed surfaces such as parking lots.
Track «TopLevelFeatureType»	A Track is a small path mainly used by pedestrians. Tracks can be segmented into Sections and Intersections.
AbstractTransportationSpace «FeatureType»	AbstractTransportationSpace is the abstract superclass of transportation objects such as Roads, Tracks, Railways, Waterways or Squares.
AuxiliaryTrafficArea «FeatureType»	An AuxiliaryTrafficArea is the ground surface of an AuxiliaryTrafficSpace.
AuxiliaryTrafficSpace «FeatureType»	An AuxiliaryTrafficSpace is a space within the transportation space not intended for traffic purposes.

ClearanceSpace «FeatureType»	A ClearanceSpace represents the actual free space above a TrafficArea within which a mobile object can move without contacting an obstruction.
Hole «FeatureType»	A Hole is an opening in the surface of a Road, Track or Square such as road damages, manholes or drains. Holes can span multiple transportation objects.
HoleSurface «FeatureType»	A HoleSurface is a representation of the ground surface of a hole.
Intersection «FeatureType»	An Intersection is a transportation space that is a shared segment of multiple Road, Track, Railway, or Waterway objects (e.g. a crossing of two roads or a level crossing of a road and a railway).
Marking «FeatureType»	A Marking is a visible pattern on a transportation area relevant to the structuring or restriction of traffic. Examples are road markings and markings related to railway or waterway traffic.
Section «FeatureType»	A Section is a transportation space that is a segment of a Road, Railway, Track, or Waterway.
TrafficArea «FeatureType»	A TrafficArea is the ground surface of a TrafficSpace. Traffic areas are the surfaces upon which traffic actually takes place.
TrafficSpace «FeatureType»	A TrafficSpace is a space in which traffic takes place. Traffic includes the movement of entities such as trains, vehicles, pedestrians, ships, or other transportation types.

Table 38. Enumerated Classes used in Transportation

Name	Description
GranularityValue «Enumeration»	GranularityValue enumerates the different levels of granularity in which transportation objects are represented.
TrafficDirectionValue «Enumeration»	TrafficDirectionValue enumerates the allowed directions of travel of a mobile object.

Table 39. CodeList Classes used in Transportation

Name	Description
AuxiliaryTrafficAreaClassValue «CodeList»	AuxiliaryTrafficAreaClassValue is a code list used to further classify an AuxiliaryTrafficArea.
AuxiliaryTrafficAreaFunctionValue «CodeList»	AuxiliaryTrafficAreaFunctionValue is a code list that enumerates the different purposes of an AuxiliaryTrafficArea.
AuxiliaryTrafficAreaUsageValue «CodeList»	AuxiliaryTrafficAreaUsageValue is a code list that enumerates the different uses of an AuxiliaryTrafficArea.

AuxiliaryTrafficSpaceClassValue «CodeList»	AuxiliaryTrafficSpaceClassValue is a code list used to further classify an AuxiliaryTrafficSpace.
AuxiliaryTrafficSpaceFunctionValue «CodeList»	AuxiliaryTrafficSpaceFunctionValue is a code list that enumerates the different purposes of an AuxiliaryTrafficSpace.
AuxiliaryTrafficSpaceUsageValue «CodeList»	AuxiliaryTrafficSpaceUsageValue is a code list that enumerates the different uses of an AuxiliaryTrafficSpace.
ClearanceSpaceClassValue «CodeList»	ClearanceSpaceClassValue is a code list used to further classify a ClearanceSpace.
HoleClassValue «CodeList»	HoleClassValue is a code list used to further classify a Hole.
IntersectionClassValue «CodeList»	IntersectionClassValue is a code list used to further classify an Intersection.
MarkingClassValue «CodeList»	MarkingClassValue is a code list used to further classify a Marking.
RailwayClassValue «CodeList»	RailwayClassValue is a code list used to further classify a Railway.
RailwayFunctionValue «CodeList»	RailwayFunctionValue is a code list that enumerates the different purposes of a Railway.
RailwayUsageValue «CodeList»	RailwayUsageValue is a code list that enumerates the different uses of a Railway.
RoadClassValue «CodeList»	RoadClassValue is a code list used to further classify a Road.
RoadFunctionValue «CodeList»	RoadFunctionValue is a code list that enumerates the different purposes of a Road.
RoadUsageValue «CodeList»	RoadUsageValue is a code list that enumerates the different uses of a Road.
SectionClassValue «CodeList»	SectionClassValue is a code list used to further classify a Section.
SquareClassValue «CodeList»	SquareClassValue is a code list used to further classify a Square.
SquareFunctionValue «CodeList»	SquareFunctionValue is a code list that enumerates the different purposes of a Square.
SquareUsageValue «CodeList»	SquareUsageValue is a code list that enumerates the different uses of a Square.
SurfaceMaterialValue «CodeList»	SurfaceMaterialValue is a code list that enumerates the different surface materials.

TrackClassValue «CodeList»	TrackClassValue is a code list used to further classify a Track.
TrackFunctionValue «CodeList»	TrackFunctionValue is a code list that enumerates the different purposes of a Track.
TrackUsageValue «CodeList»	TrackUsageValue is a code list that enumerates the different uses of a Track.
TrafficAreaClassValue «CodeList»	TrafficAreaClassValue is a code list used to further classify a TrafficArea.
TrafficAreaFunctionValue «CodeList»	TrafficAreaFunctionValue is a code list that enumerates the different purposes of a TrafficArea.
TrafficAreaUsageValue «CodeList»	TrafficAreaUsageValue is a code list that enumerates the different uses of a TrafficArea.
TrafficSpaceClassValue «CodeList»	TrafficSpaceClassValue is a code list used to further classify a TrafficSpace.
TrafficSpaceFunctionValue «CodeList»	TrafficSpaceFunctionValue is a code list that enumerates the different purposes of a TrafficSpace.
TrafficSpaceUsageValue «CodeList»	TrafficSpaceUsageValue is a code list that enumerates the different uses of a TrafficSpace.
TransportationSpaceClassValue «CodeList»	TransportationSpaceClassValue is a code list used to further classify a TransportationSpace.
TransportationSpaceFunctionValue «CodeList»	TransportationSpaceFunctionValue is a code list that enumerates the different purposes of a TransportationSpace.
TransportationSpaceUsageValue «CodeList»	TransportationSpaceUsageValue is a code list that enumerates the different uses of a TransportationSpace.
Waterway «TopLevelFeatureType»	A Waterway is a transportation space used for the movement of vessels upon or within a water body.
WaterwayClassValue «CodeList»	WaterwayClassValue is a code list used to further classify a Waterway.
WaterwayFunctionValue «CodeList»	WaterwayFunctionValue is a code list that enumerates the different purposes of a Waterway.
WaterwayUsageValue «CodeList»	WaterwayUsageValue is a code list that enumerates the different uses of a Waterway.

8.14.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model

8.15. Tunnel

Requirements Class	
http://www.opengis.net/spec/CityGML/3.1/req/req-class-tunnel	
Target type	Implementation Specification
Dependency	/req/req-class-core

The Tunnel module provides the representation of thematic and spatial aspects of tunnels. Tunnels are horizontal or sloping enclosed passage ways of a certain length, mainly underground or underwater. Tunnels are intended for passing obstacles such as mountains, waterways or other traffic routes by humans, animals or goods. Tunnels are represented in the UML model by the top-level feature type *Tunnel*, which is the main class of the Tunnel module. Tunnels can physically or functionally be subdivided into tunnel parts. In addition, tunnels can be decomposed into structural elements, such as walls, slabs, staircases, and beams.

The interior of tunnels is represented by hollow spaces. This allows a virtual accessibility of tunnels, e.g. for driving through a tunnel, for simulating disaster management or for presenting the light illumination within a tunnel. Tunnels can contain installations and furniture. Installations are permanent parts of a tunnel that strongly affect the outer or inner appearance of the tunnel and that cannot be moved. Examples are stairs, railings, radiators or pipes. Furniture, in contrast, represent moveable objects inside a tunnel, like movable equipment in control areas. Tunnels can be bounded by different types of surfaces. In this way, the outer structure of tunnels can be differentiated semantically into wall surfaces, roof surfaces, ground surfaces, outer floor surfaces, and outer ceiling surfaces, whereas the visible surface of hollow spaces can be structured into interior wall surfaces, floor surfaces, and ceiling surfaces. Furthermore, the openings of tunnels, i.e. windows and doors, can be represented including their corresponding surfaces.

The UML diagram of the Tunnel module is depicted in Tunnel UML Diagram. The Tunnel module inherits concepts from the Construction module (cf. Section Construction). The Construction module defines objects that are common to all types of construction, such as the different surface types and the openings. A detailed discussion of the Requirements Class Tunnel can be found in the CityGML Best Practices document here.

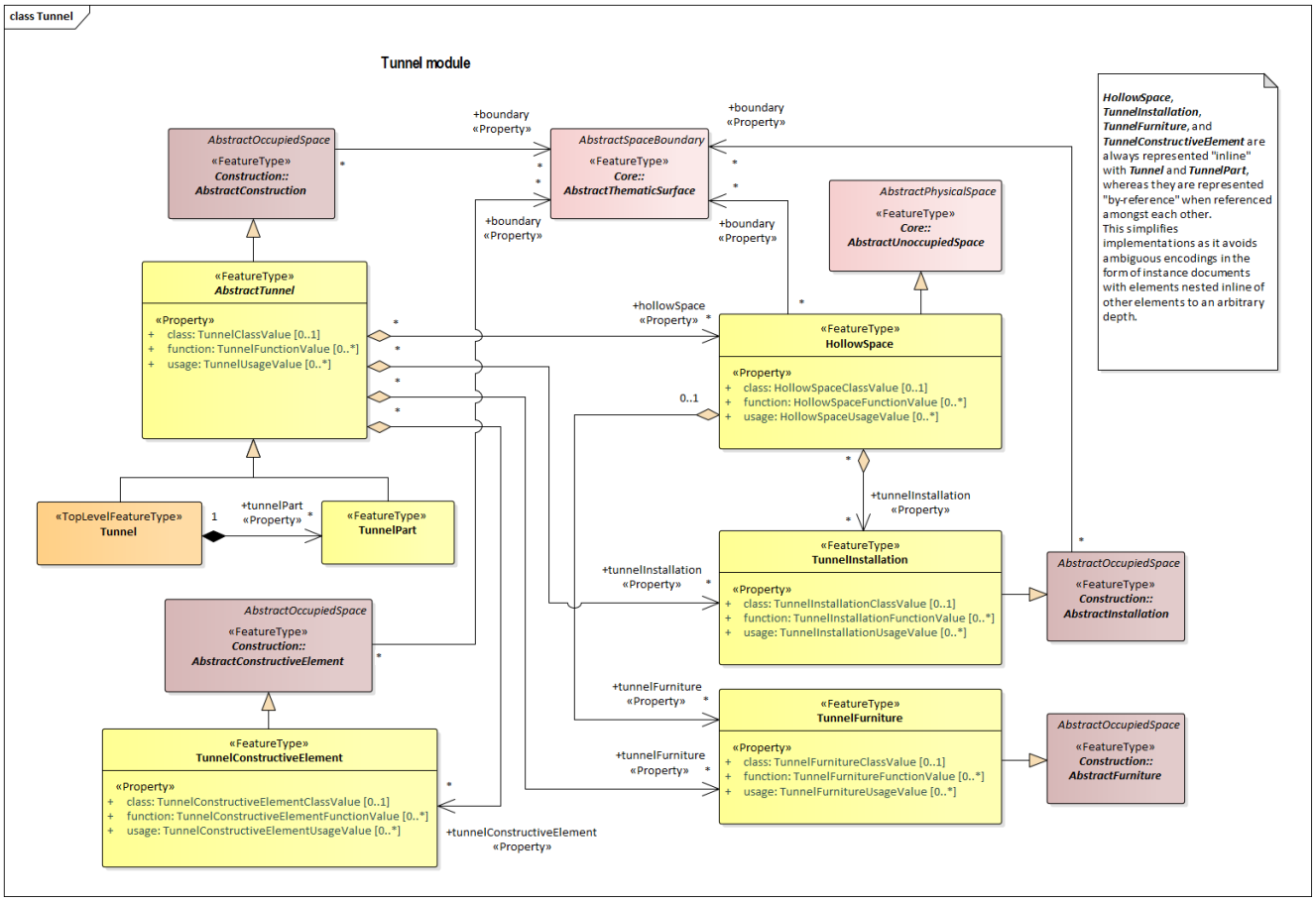

Figure 20. UML diagram of the Tunnel Model.

Table 40. Classes used in Tunnel

Class	Description
Tunnel «TopLevelFeatureType»	A Tunnel represents a horizontal or sloping enclosed passage way of a certain length, mainly underground or underwater. [cf. ISO 6707-1]
AbstractTunnel «FeatureType»	AbstractTunnel is an abstract superclass representing the common attributes and associations of the classes Tunnel and TunnelPart.
HollowSpace «FeatureType»	A HollowSpace is a space within a Tunnel or TunnelPart intended for certain functions (e.g. transport or passage ways, service rooms, emergency shelters). A HollowSpace is bounded physically and/or virtually (e.g. by ClosureSurfaces or GenericSurfaces).
TunnelConstructiveElement «FeatureType»	A TunnelConstructiveElement is an element of a Tunnel which is essential from a structural point of view. Examples are walls, slabs, beams.
TunnelFurniture «FeatureType»	A TunnelFurniture is an equipment for occupant use, usually not fixed to the tunnel. [cf. ISO 6707-1]
TunnelInstallation «FeatureType»	A TunnelInstallation is a permanent part of a Tunnel (inside and/or outside) which does not have the significance of a TunnelPart. In contrast to TunnelConstructiveElement, a TunnelInstallation is not essential from a structural point of view. Examples are stairs, antennas or railings.

TunnelPart «FeatureType»	A TunnelPart is a physical or functional subdivision of a Tunnel. It would be considered a Tunnel, if it were not part of a collection of other TunnelParts.

Table 41. CodeList Classes used in Tunnel

Name	Description
HollowSpaceClassValue «CodeList»	HollowSpaceClassValue is a code list used to further classify a HollowSpace.
HollowSpaceFunctionValue «CodeList»	HollowSpaceFunctionValue is a code list that enumerates the different purposes of a HollowSpace.
HollowSpaceUsageValue «CodeList»	HollowSpaceUsageValue is a code list that enumerates the different uses of a HollowSpace.
TunnelClassValue «CodeList»	TunnelClassValue is a code list used to further classify a Tunnel.
TunnelConstructiveElementClassValue «CodeList»	TunnelConstructiveElementClassValue is a code list used to further classify a TunnelConstructiveElement.
TunnelConstructiveElementFunctionValue «CodeList»	TunnelConstructiveElementFunctionValue is a code list that enumerates the different purposes of a TunnelConstructiveElement.
TunnelConstructiveElementUsageValue «CodeList»	TunnelConstructiveElementUsageValue is a code list that enumerates the different uses of a TunnelConstructiveElement.
TunnelFunctionValue «CodeList»	TunnelFunctionValue is a code list that enumerates the different purposes of a Tunnel.
TunnelFurnitureClassValue «CodeList»	TunnelFurnitureClassValue is a code list used to further classify a TunnelFurniture.
TunnelFurnitureFunctionValue «CodeList»	TunnelFurnitureFunctionValue is a code list that enumerates the different purposes of a TunnelFurniture.
TunnelFurnitureUsageValue «CodeList»	TunnelFurnitureUsageValue is a code list that enumerates the different uses of a TunnelFurniture.
TunnelInstallationClassValue «CodeList»	TunnelInstallationClassValue is a code list used to further classify a TunnelInstallation.
TunnelInstallationFunctionValue «CodeList»	TunnelInstallationFunctionValue is a code list that enumerates the different purposes of a TunnelInstallation.

TunnelInstallationUsageValue «CodeList»	TunnelInstallationUsageValue is a code list that enumerates the different uses of a TunnelInstallation.
TunnelUsageValue «CodeList»	TunnelUsageValue is a code list that enumerates the different uses of a Tunnel.

8.15.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.16. Vegetation

Requirements Class	
http://www.opengis.net/spec/CityGML/3.1/req/req-class-vegetation	
Target type	Implementation Specification
Dependency	/req/req-class-core

The Vegetation module defines the concepts to represent vegetation within city models. Vegetation can be represented either as solitary vegetation objects, such as trees, bushes and ferns, or as vegetation areas that are covered by plants of a given species or a typical mixture of plant species, such as forests, steppes and wet meadows. Vegetation is represented in the UML model by the top-level feature types *SolitaryVegetationObject* and *PlantCover*, which are also the only classes of the Vegetation module.

The UML diagram of the Vegetation module is depicted in Vegetation UML Diagram. A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document here.

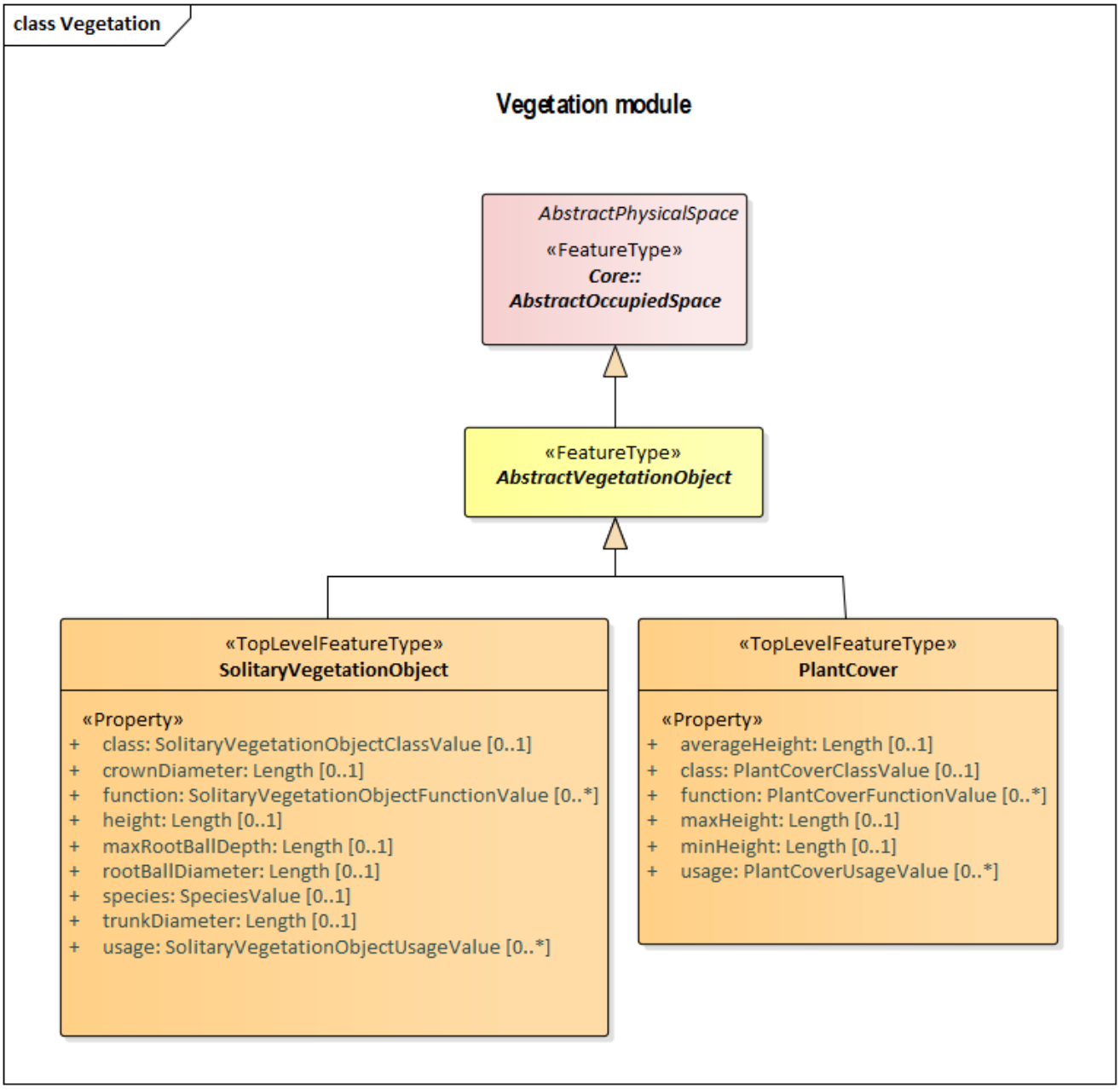

Figure 21. UML diagram of the Vegetation Model.

Table 42. Classes used in Vegetation

Class	Description
PlantCover «TopLevelFeatureType»	A PlantCover represents a space covered by vegetation.
SolitaryVegetationObject «TopLevelFeatureType»	A SolitaryVegetationObject represents individual vegetation objects, e.g. trees or bushes.
AbstractVegetationObject «FeatureType»	AbstractVegetationObject is the abstract superclass for all kinds of vegetation objects.
PlantCoverClassValue «CodeList»	PlantCoverClassValue is a code list used to further classify a PlantCover.

PlantCoverFunctionValue «CodeList»	PlantCoverFunctionValue is a code list that enumerates the different purposes of a PlantCover.
PlantCoverUsageValue «CodeList»	PlantCoverUsageValue is a code list that enumerates the different uses of a PlantCover.
SolitaryVegetationObjectClassValue «CodeList»	SolitaryVegetationObjectClassValue is a code list used to further classify a SolitaryVegetationObject.
SolitaryVegetationObjectFunctionValue «CodeList»	SolitaryVegetationObjectFunctionValue is a code list that enumerates the different purposes of a SolitaryVegetationObject.
SolitaryVegetationObjectUsageValue «CodeList»	SolitaryVegetationObjectUsageValue is a code list that enumerates the different uses of a SolitaryVegetationObject.
SpeciesValue «CodeList»	A SpeciesValue is a code list that enumerates the species of a SolitaryVegetationObject.

8.16.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.17. Versioning

Requirements Class	
http://www.opengis.net/spec/CityGML/3.0/req/req-class-versioning	
Target type	Implementation Specification
Dependency	/req/req-class-core

The Versioning module provides the concepts that allow for representing multiple versions of a city model. A specific version represents a defined state of a city model consisting of the dedicated versions of all city object instances that belong to the respective city model version. Each version can be complemented by version transitions that describe the change of the state of a city model from one version to another and that give the reason for the change and the modifications applied. In addition, the Versioning module introduces bitemporal timestamps for all objects. This allows for providing all objects with information on the time period a specific version of an object is an integral part of the 3D city model and on the lifespan a specific version of an object exists in the real world.

By using the Versioning module, slow changes over a long time period with respect to cities and city models can be represented. This includes the creation and termination of objects (e.g. construction or demolition of sites, planting of trees, construction of new roads), structural changes of objects (e.g. raising of buildings), and changes in the status of an object (e.g. change of building owner, change of the traffic direction of a road to a one-way street). In this way, the history or evolution of

cities and city models can be modelled, parallel or alternative versions of cities and city models can be managed, and changes of geometries and thematic properties of individual city objects over time can be tracked.

The UML diagram of the Versioning module is depicted in Versioning UML Diagram. A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document here.

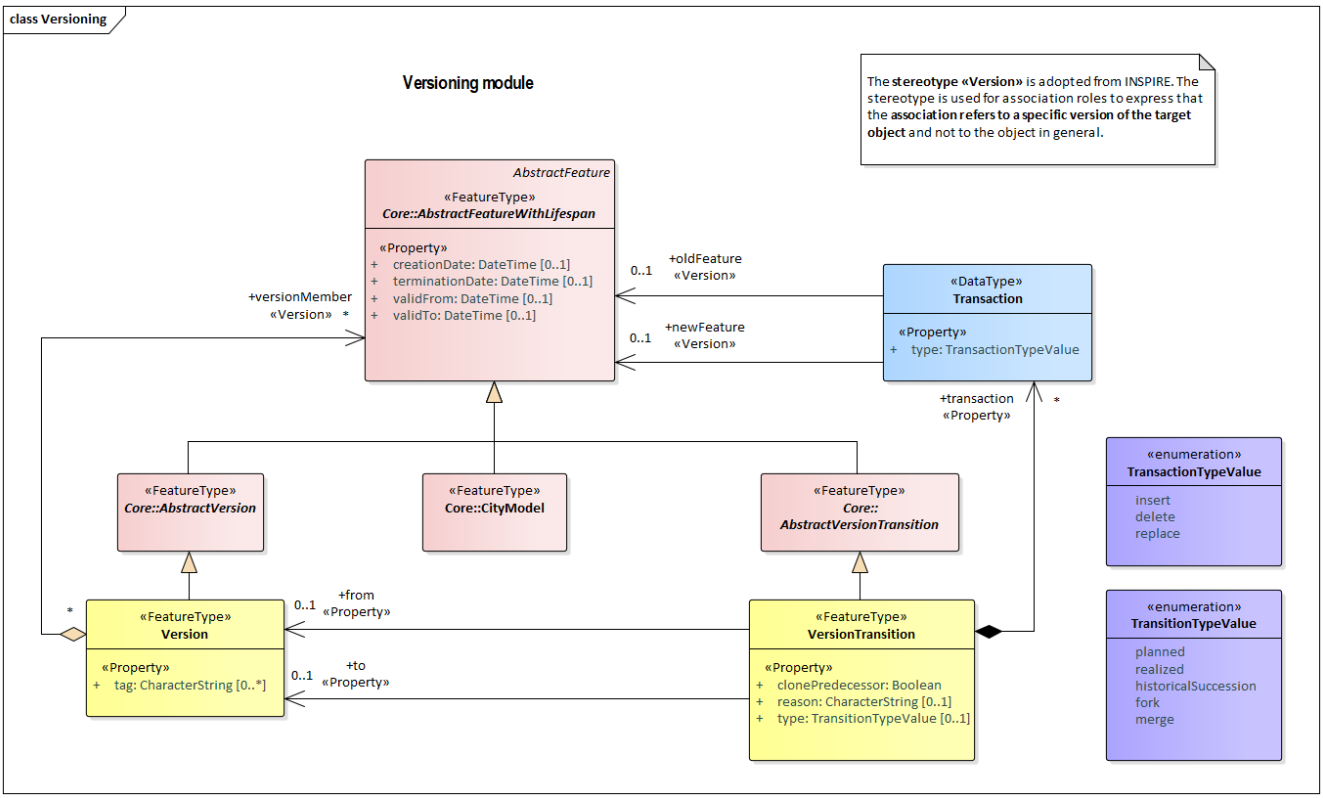

Figure 22. UML diagram of the Versioning Model.

Editing instructions:
1) Sort the entries by stereotype
a) TopLevelFeatureType
b) FeatureType
c) Object
d) DataType
e) Primitive Data Types
f) Unions
g) enumerations
h) Codelists
2) If there are a large number of entries, distribute the entries into the empty tables below based on the stereotype
3) Delete any empty tables.
4) Delete these instructions

Table 43. Classes used in Versioning

Class	Description
Version «FeatureType»	Version represents a defined state of a city model consisting of the dedicated versions of all city object instances that belong to the respective city model version. Versions can have names, a description and can be labeled with an arbitrary number of user defined tags.
VersionTransition «FeatureType»	VersionTransition describes the change of the state of a city model from one version to another. Version transitions can have names, a description and can be further qualified by a type and a reason.
Transaction «DataType»	Transaction represents a modification of the city model by the creation, termination, or replacement of a specific city object. While the creation of a city object also marks its first object version, the termination marks the end of existence of a real world object and, hence, also terminates the final version of a city object. The replacement of a city object means that a specific version of it is replaced by a new version.
TransactionTypeValue	TransactionTypeValue enumerates the three possible types of transactions: insert, delete, or replace.
TransitionTypeValue	TransitionTypeValue enumerates the different kinds of version transitions. “planned” and “fork” should be used in cases when from one city model version multiple successor versions are being created. “realized” and “merge” should be used when different city model versions are converging into a common successor version.

Table 44. Data Types used in Versioning

Name	Description

Table 45. Primitive Data Types used in Versioning

Name	Description

Table 46. Union types used in Versioning

Name	Description

Table 47. Enumerated Classes used in Versioning

Name	Description

Table 48. CodeList Classes used in Versioning

Name	Description

8.17.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

8.18. Water Body

Requirements Class	
http://www.opengis.net/spec/CityGML/3.1/req/req-class-waterbody	
Target type	Implementation Specification
Dependency	/req/req-class-core

The WaterBody module provides the representation of significant and permanent or semi-permanent accumulations of surface water, usually covering a part of the Earth. Examples of such water bodies that can be modelled with CityGML are rivers, canals, lakes, and basins. Water bodies are represented in the UML model by the top-level feature type *WaterBody*, which is the main class of the WaterBody module.

Water bodies can be bounded by water surfaces, which represent the upper exterior interface between the water body and the atmosphere, and by water ground surfaces, which represent the exterior boundary surfaces of the submerged bottom of a water body (e.g. DTM or floor of a 3D basin object). Water surfaces are dynamic surfaces, thus, the visible water surface can regularly as well as irregularly change in height and covered area due to natural forces such as tides and floods.

The UML diagram of the WaterBody module is depicted in Water Body UML Diagram. A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document here.

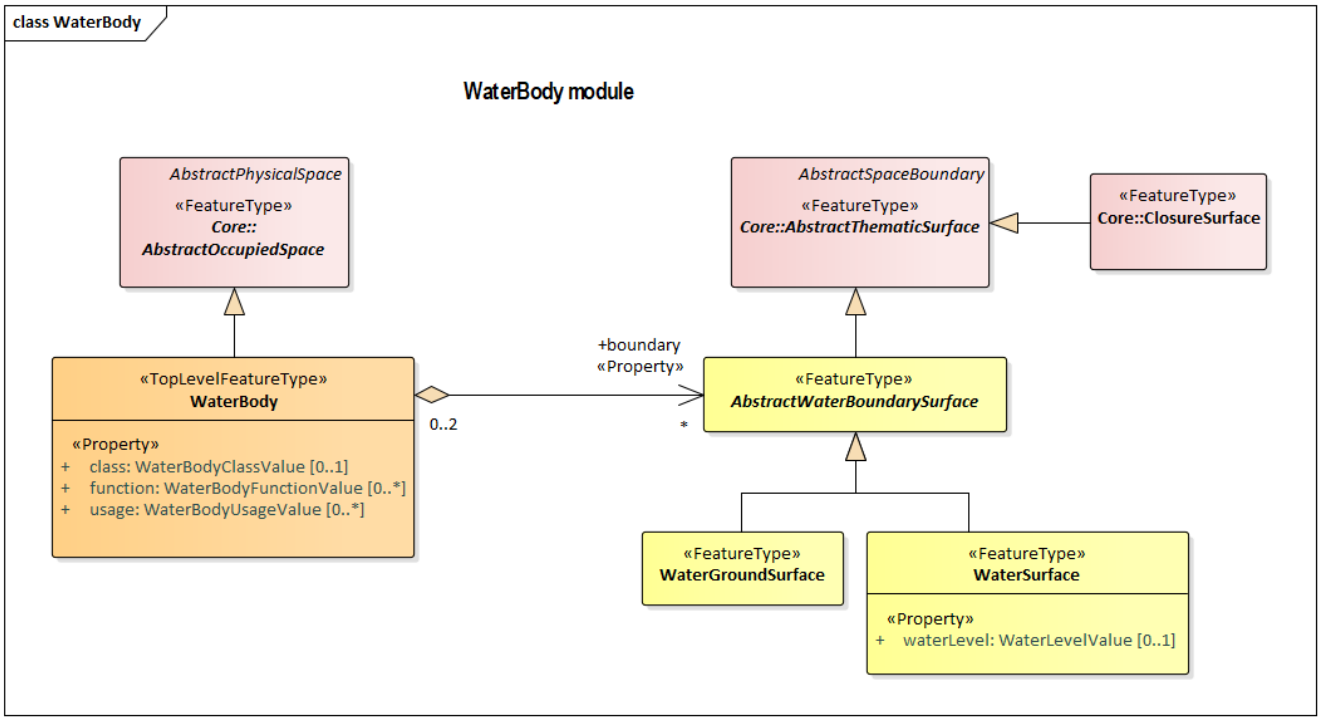

Figure 23. UML diagram of the Water Body Model.

Table 49. Classes used in WaterBody

Class	Description
WaterBody «TopLevelFeatureType»	A WaterBody represents significant and permanent or semi-permanent accumulations of surface water, usually covering a part of the Earth.

AbstractWaterBoundarySurface «FeatureType»	AbstractWaterBoundarySurface is the abstract superclass for all kinds of thematic surfaces bounding a water body.
WaterGroundSurface «FeatureType»	A WaterGroundSurface represents the exterior boundary surface of the submerged bottom of a water body.
WaterSurface «FeatureType»	A WaterSurface represents the upper exterior interface between a water body and the atmosphere.
WaterBodyClassValue «CodeList»	WaterBodyClassValue is a code list used to further classify a WaterBody.
WaterBodyFunctionValue «CodeList»	WaterBodyFunctionValue is a code list that enumerates the different purposes of a WaterBody.
WaterBodyUsageValue «CodeList»	WaterBodyUsageValue is a code list that enumerates the different uses of a WaterBody.
WaterLevelValue «CodeList»	WaterLevelValue is a code list that enumerates the different levels of a water surface.

8.18.1. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

Chapter 9. CityGML Data Dictionary

The CityGML UML model is the normative definition of the CityGML Conceptual Model. The Data Dictionary tables in this section were software generated from the UML model. As such, this section provides a normative representation of the CityGML Conceptual Model.

An alternate representation can be found in the CityGML Conceptual Model.

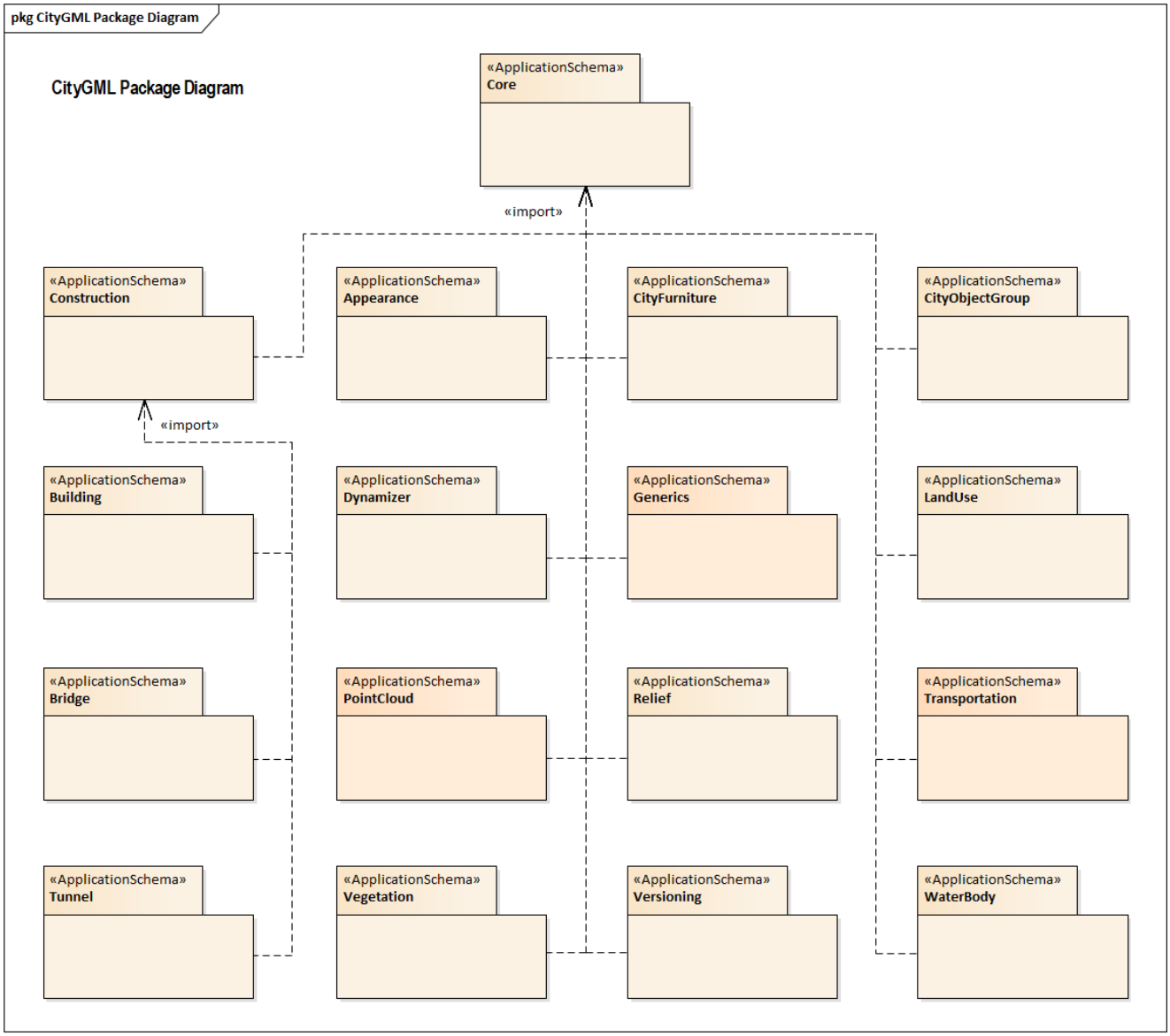

Figure 24. CityGML UML Packages

9.1. ISO Classes Data Dictionary

The following classes are defined in ISO standards and used by the CityGML Conceptual Model.

9.1.1. Classes

9.1.1.1. Class AnyFeature (ISO 19109:2015)

AnyFeature

Definition:	AnyFeature is an abstract class that is the generalization of all feature types. AnyFeature is an instance of the «metaclass» FeatureType [cf. ISO 19109].
Subclass Of:	None
StereoType:	«FeatureType»

9.1.1.2. Class CV_DiscreteGridPointCoverage (ISO 19123:2005)

CV_DiscreteGridPointCoverage

Definition:	
Subclass Of:	CV_DiscreteCoverage
StereoType:	«type»

Role name	Target class and multiplicity	Definition
valueAssignment	CV_GridValuesMatrix [1]	
element	CV_GridPointValuePair [1..*]	

9.1.1.3. Class GM_Object

GM_Object

Definition:	GM_Object is the root class of the geometric object taxonomy and supports interfaces common to all geographically referenced geometric objects. GM_Object instances are sets of direct positions in a particular coordinate reference system. A GM_Object can be regarded as an infinite set of points that satisfies the set operation interfaces for a set of direct positions, TransfiniteSet<DirectPosition>. Since an infinite collection class cannot be implemented directly, a Boolean test for inclusion shall be provided by the GM_Object interface. This international standard concentrates on vector geometry classes, but future work may use GM_Object as a root class without modification. NOTE As a type, GM_Object does not have a well-defined default state or value representation as a data type. Instantiated subclasses of GM_Object will.
Subclass Of:	None
StereoType:	«type»
Constraint:	dimension() > boundary().dimension (Invariant):
Constraint:	boundary().notEmpty() implies boundary().dimension() = dimension() -1 (Invariant):
Constraint:	boundary().isEmpty() = isCycle() (Invariant):

Role name	Target class and multiplicity	Definition
CRS	CRS [0..1]	The association "CRS" links this GM_Object to the coordinate reference system, as defined in ISO150456-9, used in its DirectPosition coordinates GM_Object::CRS : CRS NOTE This association shall only be navigable from GM_Object to CRS. This means that the coordinate reference system objects in a data set do not keep a list of GM_Objects that use them. A detailed description of reference systems may be found in ISO 15046 Part 11.
CRS	SC_CRS [0..1]	The association "CRS" links this GM_Object to the coordinate reference system, as defined in ISO150456-9, used in its DirectPosition coordinates GM_Object::CRS : CRS NOTE This association shall only be navigable from GM_Object to CRS. This means that the coordinate reference system objects in a data set do not keep a list of GM_Objects that use them. A detailed description of reference systems may be found in ISO 15046 Part 11.
	CV_DomainObject []	
	Geometry []	

9.1.1.4. Class GM_MultiCurve (ISO 19107: 2003)

GM_MultiCurve	
Definition:	
Subclass Of:	GM_MultiPrimitive
StereoType:	«type»

9.1.1.5. Class GM_MultiSurface (ISO 19107:2003)

GM_MultiSurface	
Definition:	
Subclass Of:	GM_MultiPrimitive
StereoType:	«type»

Attribute	Value type and multiplicity	Definition
area	Area	
perimeter	Length	

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.1.6. Class GM_Point (ISO 19107:2003)

GM_Point	
Definition:	GM_Point (Figure 9) is the basic data type for a geometric object consisting of one and only one point.
Subclass Of:	GM_Primitive
StereoType:	«type»

Role name	Target class and multiplicity	Definition
	Point []	
composite	GM_CompositePoint [0..*]	

Attribute	Value type and multiplicity	Definition
position	DirectPosition	The attribute "position" shall be the DirectPosition of this GM_Point. GM_Point::position [1] : DirectPosition NOTE In most cases, the state of a GM_Point is fully determined by its position attribute. The only exception to this is if the GM_Point has been subclassed to provide additional non-geometric information such as symbology.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.1.7. Class GM_Solid (ISO 19107:2003)

GM_Solid

Definition:	GM_Solid (Figure 13), a subclass of GM_Primitive, is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces.
Subclass Of:	GM_Primitive
StereoType:	«type»

Role name	Target class and multiplicity	Definition
composite	GM_CompositeSolid [0..*] Solid []	

9.1.1.8. Class GM_Surface (ISO 19107:2003)

GM_Surface

Definition:	GM_Surface (Figure 12) a subclass of GM_Primitive and is the basis for 2-dimensional geometry. Unorientable surfaces such as the Möbius band are not allowed. The orientation of a surface chooses an "up" direction through the choice of the upward normal, which, if the surface is not a cycle, is the side of the surface from which the exterior boundary appears counterclockwise. Reversal of the surface orientation reverses the curve orientation of each boundary component, and interchanges the conceptual "up" and "down" direction of the surface. If the surface is the boundary of a solid, the "up" direction is usually outward. For closed surfaces, which have no boundary, the up direction is that of the surface patches, which must be consistent with one another. Its included GM_SurfacePatches describe the interior structure of a GM_Surface. NOTE Other than the restriction on orientability, no other "validity" condition is required for GM_Surface.
Subclass Of:	GM_OrientableSurface
StereoType:	«type»

Role name	Target class and multiplicity	Definition
	GM_GenericSurface [1..1]	
	Building [0..*]	

9.1.1.9. Class GM_Tin (ISO 19107:2003)

GM_Tin

Definition:	A GM_Tin (Figure 21) is a GM_TriangulatedSurface that uses the Delaunay algorithm or a similar algorithm complemented with consideration for breaklines, stoplines and maximum length of triangle sides (Figure 22). These networks satisfy the Delaunay criterion away from the modifications: For each triangle in the network, the circle passing through its vertexes does not contain, in its interior, the vertex of any other triangle.
Subclass Of:	GM_TriangulatedSurface
StereoType:	«type»

Attribute	Value type and multiplicity	Definition
breakLines	Set<GM_LineString>	
controlPoint	GM_Position [3..*]	
maxLength	Distance	
stopLines	Set<GM_LineString>	

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.1.10. Class GM_TriangulatedSurface (ISO 19107:2003)

GM_TriangulatedSurface

Definition:	A GM_TriangulatedSurface (Figure 21) is a GM_PolyhedralSurface that is composed only of triangles (GM_Triangle). There is no restriction on how the triangulation is derived.
Subclass Of:	GM_PolyhedralSurface
StereoType:	«type»

9.1.1.11. Class SC_CRS (ISO 19111:2019)

SC_CRS

Definition:	Coordinate reference system which is usually single but may be compound.
Subclass Of:	IO_IdentifiedObjectBase, RS_ReferenceSystem
StereoType:	«type»

Role name	Target class and multiplicity	Definition
grid	CV_ReferenceableGrid [0..*]	
coordOperationTo	CC_CoordinateOperation [0..*]	The "sourceCRS" and "targetCRS" associations are mandatory for coordinate transformations only. Coordinate conversions have a source CRS and a target CRS that are NOT specified through these associations, but through associations from GeneralDerivedCRS to SingleCRS.

Attribute	Value type and multiplicity	Definition
scope	CharacterString [1..*]	Description of usage, or limitations of usage, for which this CRS is valid. If unknown, enter "not known".

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.2. Package

Package Core

Description:	The Core module defines the basic components of the CityGML data model. The Core module defines abstract base classes that define the core properties of more specialized thematic classes defined in other modules. The Core module also defines concrete classes that are common to other modules, for example basic data types.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.3. Classes

9.1.3.1. Class AbstractAppearance

AbstractAppearance

Definition:	AbstractAppearance is the abstract superclass to represent any kind of appearance objects.
Subclass Of:	AbstractFeatureWithLifespan
StereoType:	«FeatureType»

9.1.3.2. Class AbstractCityObject

AbstractCityObject

Definition:	AbstractCityObject is the abstract superclass of all thematic classes within the CityGML data model.
Subclass Of:	AbstractFeatureWithLifespan
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
appearance	AbstractAppearance [*]	
dynamizer	AbstractDynamizer [*]	
generalizesTo	AbstractCityObject [*]	
relatedTo	AbstractCityObject [*]	
genericAttribute	AbstractGenericAttribute [*]	
externalReference	ExternalReference [*]	

Attribute	Value type and multiplicity	Definition
relativeToTerrain	RelativeToTerrain [0..1]	Describes the vertical position of the city object relative to the surrounding terrain.
relativeToWater	RelativeToWater [0..1]	Describes the vertical position of the city object relative to the surrounding water surface.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.3. Class AbstractDynamizer

AbstractDynamizer

Definition:	AbstractDynamizer is the abstract superclass to represent Dynamizer objects.
Subclass Of:	AbstractFeatureWithLifespan
StereoType:	«FeatureType»

9.1.3.4. Class AbstractFeature

AbstractFeature

Definition:	AbstractFeature is the abstract superclass of all feature types within the CityGML data model.
Subclass Of:	AnyFeature
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
description	CharacterString [0..1]	Provides further information on the feature.
featureID	ID	Specifies the unique identifier of the feature that is valid in the instance document within which it occurs.
identifier	ScopedName [0..1]	Specifies the unique identifier of the feature that is valid globally.
name	GenericName [0..*]	Specifies the name of the feature.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.5. Class AbstractFeatureWithLifespan

AbstractFeatureWithLifespan

Definition:	AbstractFeatureWithLifespan is the base class for all CityGML features. It allows the optional specification of the real-world and database times for the existence of each feature.
Subclass Of:	AbstractFeature
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
creationDate	DateTime [0..1]	Indicates the date at which a CityGML feature was added to the CityModel.
terminationDate	DateTime [0..1]	Indicates the date at which a CityGML feature was removed from the CityModel.
validFrom	DateTime [0..1]	Indicates the date at wich a CityGML feature started to exist in the real world.
validTo	DateTime [0..1]	Indicates the date at wich a CityGML feature ended to exist in the real world.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.6. Class AbstractLogicalSpace

AbstractLogicalSpace	
Definition:	AbstractLogicalSpace is the abstract superclass for all types of logical spaces. Logical space refers to spaces that are not bounded by physical surfaces but are defined according to thematic considerations.
Subclass Of:	AbstractSpace
StereoType:	«FeatureType»

9.1.3.7. Class AbstractOccupiedSpace

AbstractOccupiedSpace	
Definition:	AbstractOccupiedSpace is the abstract superclass for all types of physically occupied spaces. Occupied space refers to spaces that are partially or entirely filled with matter.
Subclass Of:	AbstractPhysicalSpace
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
lod1ImplicitRepresentation	ImplicitGeometry [0..1]	
lod2ImplicitRepresentation	ImplicitGeometry [0..1]	
lod3ImplicitRepresentation	ImplicitGeometry [0..1]	

9.1.3.8. Class AbstractPhysicalSpace

AbstractPhysicalSpace	
Definition:	AbstractPhysicalSpace is the abstract superclass for all types of physical spaces. Physical space refers to spaces that are fully or partially bounded by physical objects.
Subclass Of:	AbstractSpace
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
lod3TerrainIntersectionCurve	GM_MultiCurve [0..1]	
pointCloud	AbstractPointCloud [0..1]	
lod1TerrainIntersectionCurve	GM_MultiCurve [0..1]	
lod2TerrainIntersectionCurve	GM_MultiCurve [0..1]	

9.1.3.9. Class AbstractPointCloud

AbstractPointCloud	
Definition:	AbstractPointCloud is the abstract superclass to represent PointCloud objects.
Subclass Of:	AbstractFeature
StereoType:	«FeatureType»

9.1.3.10. Class AbstractSpace

AbstractSpace	
Definition:	AbstractSpace is the abstract superclass for all types of spaces. A space is an entity of volumetric extent in the real world.
Subclass Of:	AbstractCityObject
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
lod2Solid	GM_Solid [0..1]	
boundary	AbstractSpaceBoundary [*]	
lod3MultiCurve	GM_MultiCurve [0..1]	
lod1Solid	GM_Solid [0..1]	
lod0MultiSurface	GM_MultiSurface [0..1]	
lod0MultiCurve	GM_MultiCurve [0..1]	
lod0Point	GM_Point [0..1]	
lod3Solid	GM_Solid [0..1]	
lod3MultiSurface	GM_MultiSurface [0..1]	
lod2MultiSurface	GM_MultiSurface [0..1]	
lod2MultiCurve	GM_MultiCurve [0..1]	

Attribute	Value type and multiplicity	Definition
area	QualifiedArea [0..*]	Specifies qualified areas related to the space.
spaceType	SpaceType [0..1]	Specifies the degree of openness of a space.
volume	QualifiedVolume [0..*]	Specifies qualified volumes related to the space.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.11. Class AbstractSpaceBoundary

AbstractSpaceBoundary

Definition:	AbstractSpaceBoundary is the abstract superclass for all types of space boundaries. A space boundary is an entity with areal extent in the real world. Space boundaries are objects that bound a Space. They also realize the contact between adjacent spaces.
Subclass Of:	AbstractCityObject
StereoType:	«FeatureType»

9.1.3.12. Class AbstractThematicSurface

AbstractThematicSurface

Definition:	AbstractThematicSurface is the abstract superclass for all types of thematic surfaces.
Subclass Of:	AbstractSpaceBoundary
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
lod0MultiCurve	GM_MultiCurve [0..1]	
lod0MultiSurface	GM_MultiSurface [0..1]	
lod3MultiSurface	GM_MultiSurface [0..1]	
lod1MultiSurface	GM_MultiSurface [0..1]	
pointCloud	AbstractPointCloud [0..1]	
lod2MultiSurface	GM_MultiSurface [0..1]	

Attribute	Value type and multiplicity	Definition
area	QualifiedArea [0..*]	Specifies qualified areas related to the thematic surface.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.13. Class AbstractUnoccupiedSpace

AbstractUnoccupiedSpace

Definition:	AbstractUnoccupiedSpace is the abstract superclass for all types of physically unoccupied spaces. Unoccupied space refers to spaces that are entirely or mostly free of matter.
Subclass Of:	AbstractPhysicalSpace
StereoType:	«FeatureType»

9.1.3.14. Class AbstractVersion

AbstractVersion

Definition:	AbstractVersion is the abstract superclass to represent Version objects.
Subclass Of:	AbstractFeatureWithLifespan
StereoType:	«FeatureType»

9.1.3.15. Class AbstractVersionTransition

AbstractVersionTransition

Definition:	AbstractVersionTransition is the abstract superclass to represent VersionTransition objects.
Subclass Of:	AbstractFeatureWithLifespan
StereoType:	«FeatureType»

9.1.3.16. Class Address

Address

Definition:	Address represents an address of a city object.
Subclass Of:	AbstractFeature
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
multiPoint	GM_MultiPoint [0..1]	
xalAddress	XALAddressDetails [1]	

9.1.3.17. Class CityModel

CityModel

Definition:	CityModel is the container for all objects belonging to a city model.
Subclass Of:	AbstractFeatureWithLifespan
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
cityModelMember	CityModelMember [*]	

Attribute	Value type and multiplicity	Definition
engineeringCRS	EngineeringCRS [0..1]	Specifies the local coordinate reference system of the CityModel.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.18. Class CityObjectRelation

CityObjectRelation

Definition:	CityObjectRelation represents a specific relation from the city object in which it is included to another city object.
Subclass Of:	None
StereoType:	«ObjectType»

Role name	Target class and multiplicity	Definition
genericAttribute	AbstractGenericAttribute [*]	

Attribute	Value type and multiplicity	Definition
relationType	RelationTypeValue	Indicates the specific type of the CityObjectRelation.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.19. Class ClosureSurface

ClosureSurface	
Definition:	ClosureSurface is a special type of thematic surface used to close holes in volumetric objects. Closure surfaces are virtual (non-physical) surfaces.
Subclass Of:	AbstractThematicSurface
StereoType:	«FeatureType»

9.1.3.20. Class DoubleBetween0and1

DoubleBetween0and1	
Definition:	DoubleBetween0and1 is a basic type for values, which are greater or equal than 0 and less or equal than 1. The type is used for color encoding, for example.
Subclass Of:	None
StereoType:	«BasicType»
Constraint:	valueBetween0and1 (OCL): inv: DoubleBetween0and1.allInstances()→forAll(p | p >=0 and p ⇐ 1)

9.1.3.21. Class DoubleBetween0and1List

DoubleBetween0and1List	
Definition:	DoubleBetween0and1List is a basic type that represents a list of double values greater or equal than 0 and less or equal than 1. The type is used for color encoding, for example.
Subclass Of:	None
StereoType:	«BasicType»

Attribute	Value type and multiplicity	Definition
list	DoubleBetween0and1	Specifies the list of double values.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.22. Class DoubleList

DoubleList

Definition:	DoubleList is an ordered sequence of double values.
Subclass Of:	None
StereoType:	«BasicType»

Attribute	Value type and multiplicity	Definition
list	Real	Specifies the list of double values.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.23. Class DoubleOrNilReasonList

DoubleOrNilReasonList

Definition:	DoubleOrNilReasonList is a basic type that represents a list of double values and/or nil reasons.
Subclass Of:	None
StereoType:	«BasicType»

Attribute	Value type and multiplicity	Definition
list	DoubleOrNilReason	Specifies the list of double values and/or nil reasons.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.24. Class ID

ID

Definition:	ID is a basic type that represents a unique identifier.
Subclass Of:	None
StereoType:	«BasicType»

9.1.3.25. Class ImplicitGeometry

ImplicitGeometry

Definition:	ImplicitGeometry is a geometry representation 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 can be re-used or referenced many times, wherever the corresponding feature occurs in the 3D city model.
Subclass Of:	None
StereoType:	«ObjectType»

Role name	Target class and multiplicity	Definition
appearance	AbstractAppearance [*]	
relativeGeometry	GM_Object [0..1]	
referencePoint	GM_Point [1]	

Attribute	Value type and multiplicity	Definition
libraryObject	URI [0..1]	Specifies the URI that points to the prototypical geometry stored in an external file.
mimeType	MimeTypeValue [0..1]	Specifies the MIME type of the external file that stores the prototypical geometry.
objectID	ID	Specifies the unique identifier of the ImplicitGeometry.
transformationMatrix	TransformationMatrix4x4	Specifies the mathematical transformation (translation, rotation, and scaling) between the prototypical geometry and the actual spatial position of the object.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.26. Class IntegerBetween0and3

IntegerBetween0and3

Definition:	IntegerBetween0and3 is a basic type for integer values, which are greater or equal than 0 and less or equal than 3. The type is used for encoding the LOD number.
Subclass Of:	None
StereoType:	«BasicType»
Constraint:	valueBetween0and4 (OCL): inv: IntegerBetween0and4.allInstances()→forAll(p | p >= 0 and p ⇐ 4)

9.1.3.27. Class IntervalValue

IntervalValue	
Definition:	IntervalValue is a code list used to specify a time period.
Subclass Of:	None
StereoType:	«CodeList»

9.1.3.28. Class MeasureOrNilReasonList

MeasureOrNilReasonList	
Definition:	MeasureOrNilReasonList is a basic type that represents a list of double values and/or nil reasons together with a unit of measurement.
Subclass Of:	DoubleOrNilReasonList
StereoType:	«BasicType»

Attribute	Value type and multiplicity	Definition
uom	UnitOfMeasure	Specifies the unit of measurement of the double values.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.29. Class MimeTypeValue

MimeTypeValue	
Definition:	MimeTypeValue is a code list used to specify the MIME type of a referenced resource.
Subclass Of:	None
StereoType:	«CodeList»

9.1.3.30. Class NilReasonEnumeration

NilReasonEnumeration	
Definition:	NilReasonEnumeration is a code list that enumerates the different nil reasons.
Subclass Of:	None
StereoType:	«CodeList»

9.1.3.31. Class OccupantTypeValue

OccupantTypeValue	
Definition:	OccupantTypeValue is a code list used to classify occupants.
Subclass Of:	None
StereoType:	«CodeList»

9.1.3.32. Class OtherRelationTypeValue

OtherRelationTypeValue	
Definition:	OtherRelationTypeValue is a code list used to classify other types of city object relations.
Subclass Of:	RelationTypeValue
StereoType:	«CodeList»

9.1.3.33. Class QualifiedAreaTypeValue

QualifiedAreaTypeValue	
Definition:	QualifiedAreaTypeValue is a code list used to specify area types.
Subclass Of:	None
StereoType:	«CodeList»

9.1.3.34. Class QualifiedVolumeTypeValue

QualifiedVolumeTypeValue

Definition:	QualifiedVolumeTypeValue is a code list used to specify volume types.
Subclass Of:	None
StereoType:	«CodeList»

9.1.3.35. Class RelationTypeValue

RelationTypeValue	
Definition:	RelationTypeValue is a code list used to classify city object relations.
Subclass Of:	None
StereoType:	«CodeList»

9.1.3.36. Class TemporalRelationTypeValue

TemporalRelationTypeValue	
Definition:	TemporalRelationTypeValue is a code list used to classify temporal city object relations.
Subclass Of:	RelationTypeValue
StereoType:	«CodeList»

9.1.3.37. Class TopologicRelationTypeValue

TopologicRelationTypeValue	
Definition:	TopologicRelationTypeValue is a code list used to classify topological city object relations.
Subclass Of:	RelationTypeValue
StereoType:	«CodeList»

9.1.3.38. Class TransformationMatrix2x2

TransformationMatrix2x2

Definition:	TransformationMatrix2x2 is a 2 by 2 matrix represented as a list of four double values in row major order.
Subclass Of:	DoubleList
StereoType:	«BasicType»
Constraint:	lengthOfList (OCL): inv: self.list→size() = 4

9.1.3.39. Class TransformationMatrix3x4

TransformationMatrix3x4	
Definition:	TransformationMatrix3x4 is a 3 by 4 matrix represented as a list of twelve double values in row major order.
Subclass Of:	DoubleList
StereoType:	«BasicType»
Constraint:	lengthOfList (OCL): inv: self.list→size() = 12

9.1.3.40. Class TransformationMatrix4x4

TransformationMatrix4x4	
Definition:	TransformationMatrix4x4 is a 4 by 4 matrix represented as a list of sixteen double values in row major order.
Subclass Of:	DoubleList
StereoType:	«BasicType»
Constraint:	lengthOfList (OCL): inv: self.list→size() = 16

9.1.3.41. Class AbstractGenericAttribute

AbstractGenericAttribute	
Definition:	AbstractGenericAttribute is the abstract superclass for all types of generic attributes.
Subclass Of:	None
StereoType:	«DataType»

9.1.3.42. Class CityModelMember

CityModelMember

Definition:	CityModelMember is a union type that enumerates the different types of objects that can occur as members of a city model.
Subclass Of:	None
StereoType:	«Union»

Attribute	Value type and multiplicity	Definition
appearanceMember	AbstractAppearance	Specifies the appearances of the CityModel.
cityObjectMember	AbstractCityObject	Specifies the city objects that are part of the CityModel.
featureMember	AbstractFeature	Specifies the feature objects that are part of the CityModel. It allows to include objects that are not derived from a class defined in the CityGML data model, but from the ISO 19109 class AnyFeature.
versionMember	AbstractVersion	Specifies the different versions of the CityModel.
versionTransitionMember	AbstractVersionTransition	Specifies the transitions between the different versions of the CityModel.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.43. Class DoubleOrNilReason

DoubleOrNilReason

Definition:	DoubleOrNilReason is a union type that allows for choosing between a double value and a nil reason.
Subclass Of:	None
StereoType:	«Union»

Attribute	Value type and multiplicity	Definition
nilReason	NilReason	Specifies the nil reason.
value	Real	Specifies the double value.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.44. Class ExternalReference

ExternalReference	
Definition:	ExternalReference is a reference to a corresponding object in another information system, for example in the German cadastre (ALKIS), the German topographic information system (ATKIS), or the OS UK MasterMap®.
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
informationSystem	URI [0..1]	Specifies the URI that points to the external information system.
relationType	URI [0..1]	Specifies an URI that additionally qualifies the ExternalReference. The URI can point to a definition from an external ontology (e.g. the sameAs relation from OWL) and allows for mapping the ExternalReference to RDF triples.
targetResource	URI	Specifies the URI that points to the object in the external information system.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.45. Class NilReason

NilReason	
Definition:	NilReason is a union type that allows for choosing between two different types of nil reason.
Subclass Of:	None
StereoType:	«Union»

Attribute	Value type and multiplicity	Definition
nilReasonEnumeration	NilReasonEnumeration	Indicates a nil reason that is provided in a code list.
URI	URI	Specifies a URI that points to a resource that describes the nil reason.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.46. Class Occupancy

Occupancy	
Definition:	Occupancy is an application-dependent indication of what is contained by a feature.
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
interval	IntervalValue [0..1]	Indicates the time period the occupants are contained by a feature.
numberOfOccupants	Integer	Indicates the number of occupants contained by a feature.
occupantType	OccupantTypeValue [0..1]	Indicates the specific type of the occupants that are contained by a feature.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.47. Class QualifiedArea

QualifiedArea	
Definition:	QualifiedArea is an application-dependent measure of the area of a space or of a thematic surface.
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
area	Area	Specifies the value of the QualifiedArea.
typeOfArea	QualifiedAreaTypeValue	Indicates the specific type of the QualifiedArea.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.48. Class QualifiedVolume

QualifiedVolume

Definition:	QualifiedVolume is an application-dependent measure of the volume of a space.
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
typeOfVolume	QualifiedVolumeTypeValue	Indicates the specific type of the QualifiedVolume.
volume	Volume	Specifies the value of the QualifiedVolume.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.49. Class RelativeToTerrain

RelativeToTerrain

Definition:	RelativeToTerrain enumerates the spatial relations of a city object relative to terrain in a qualitative way.
Subclass Of:	None
StereoType:	

Attribute	Value type and multiplicity	Definition
entirelyAboveTerrain		Indicates that the city object is located entirely above the terrain.
substantiallyAboveTerrain		Indicates that the city object is for the most part located above the terrain.
substantiallyAboveAndBelowTerrain		Indicates that the city object is located half above the terrain and half below the terrain.
substantiallyBelowTerrain		Indicates that the city object is for the most part located below the terrain.
entirelyBelowTerrain		Indicates that the city object is located entirely below the terrain.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.50. Class RelativeToWater

RelativeToWater

Definition:	RelativeToWater enumerates the spatial relations of a city object relative to the water surface in a qualitative way.
Subclass Of:	None
StereoType:	

Attribute	Value type and multiplicity	Definition
entirelyAboveWaterSurface		Indicates that the city object is located entirely above the water surface.
substantiallyAboveWaterSurface		Indicates that the city object is for the most part located above the water surface.
substantiallyAboveAndBelowWaterSurface		Indicates that the city object is located half above the water surface and half below the water surface.
substantiallyBelowWaterSurface		Indicates that the city object is for the most part located below the water surface.
entirelyBelowWaterSurface		Indicates that the city object is located entirely below the water surface.
temporarilyAboveAndBelowWaterSurface		Indicates that city object is temporarily located above or below the water level, because the height of the water surface is varying.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.51. Class SpaceType

SpaceType

Definition:	SpaceType is an enumeration that characterises a space according to its closure properties.
Subclass Of:	None
StereoType:	

Attribute	Value type and multiplicity	Definition
closed		Indicates that the space has boundaries at the bottom, at the top, and on all sides.
open		Indicates that the space has at maximum a boundary at the bottom.
semiOpen		Indicates that the space has a boundary at the bottom and on at least one side.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.3.52. Class XALAddressDetails

XALAddressDetails	
Definition:	XALAddressDetails represents address details according to the OASIS xAL standard.
Subclass Of:	None
StereoType:	«DataType»

9.1.4. Package

Package Appearance	
Description:	The Appearance module supports the modelling of the observable surface properties of CityGML features in the form of textures and material.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.5. Classes

9.1.5.1. Class AbstractSurfaceData

AbstractSurfaceData	
Definition:	AbstractSurfaceData is the abstract superclass for different kinds of textures and material.
Subclass Of:	AbstractFeature
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
isFront	Boolean [0..1]	Indicates whether the texture or material is assigned to the front side or the back side of the surface geometry object.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.5.2. Class AbstractTexture

AbstractTexture

Definition:	AbstractTexture is the abstract superclass to represent the common attributes of the classes ParameterizedTexture and GeoreferencedTexture.
Subclass Of:	AbstractSurfaceData
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
borderColor	ColorPlusOpacity [0..1]	Specifies the color of that part of the surface that is not covered by the texture.
imageURI	URI	Specifies the URI that points to the external image data file.
mimeType	MimeTypeValue [0..1]	Specifies the MIME type of the external point cloud file.
textureType	TextureType [0..1]	Indicates the specific type of the texture.
wrapMode	WrapMode [0..1]	Specifies the behaviour of the texture when the texture is smaller than the surface to which it is applied.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.5.3. Class Appearance

Appearance

Definition:	An Appearance is a collection of surface data, i.e. observable properties for surface geometry objects in the form of textures and material.
Subclass Of:	AbstractAppearance
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
surfaceData	AbstractSurfaceData [*]	

Attribute	Value type and multiplicity	Definition
theme	CharacterString [0..1]	Specifies the topic of the Appearance. Each Appearance contains surface data for one theme only. Examples of themes are infrared radiation, noise pollution, or earthquake-induced structural stress.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.5.4. Class Color

Color

Definition:	Color is a list of three double values between 0 and 1 defining an RGB color value.
Subclass Of:	DoubleBetween0and1List
StereoType:	«BasicType»
Constraint:	lengthOfList (OCL): inv: self.list→size() = 3

9.1.5.5. Class ColorPlusOpacity

ColorPlusOpacity

Definition:	Color is a list of four double values between 0 and 1 defining an RGBA color value. Opacity value of 0 means transparent.
Subclass Of:	DoubleBetween0and1List
StereoType:	«BasicType»
Constraint:	lengthOfList (OCL): inv: self.list→size() = 3 or self.list→size() = 4

9.1.5.6. Class GeoreferencedTexture

GeoreferencedTexture

Definition:	A GeoreferencedTexture is a texture that uses a planimetric projection. It contains an implicit parameterization that is either stored within the image file, an acompanying world file or specified using the orientation and referencePoint elements.
Subclass Of:	AbstractTexture
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
referencePoint	GM_Point [0..1]	

Attribute	Value type and multiplicity	Definition
orientation	TransformationMatrix2x2 [0..1]	Specifies the rotation and scaling of the image in form of a 2x2 matrix.
preferWorldFile	Boolean [0..1]	Indicates whether the georeference from the image file or the accompanying world file should be prefered.
target	URI [0..*]	Specifies the URI that points to the surface geometry objects to which the texture is applied.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.5.7. Class ParameterizedTexture

ParameterizedTexture

Definition:	A ParameterizedTexture is a texture that uses texture coordinates or a transformation matrix for parameterization.
Subclass Of:	AbstractTexture
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
textureParameterization	AbstractTextureParameterization [*]	

9.1.5.8. Class TextureAssociation

TextureAssociation

Definition:	TextureAssociation denotes the relation of a texture to a surface geometry object.
Subclass Of:	None
StereoType:	«ObjectType»

[

Attribute	Value type and multiplicity	Definition
target	URI	Specifies the URI that points to the surface geometry object to which the texture is applied.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.5.9. Class X3DMaterial

X3DMaterial

Definition:	X3DMaterial defines properties for surface geometry objects based on the material definitions from the standards X3D and COLLADA.
Subclass Of:	AbstractSurfaceData
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
ambientIntensity	DoubleBetween0and1 [0..1]	Specifies the minimum percentage of diffuseColor that is visible regardless of light sources.
diffuseColor	Color [0..1]	Specifies the color of the light diffusely reflected by the surface geometry object.
emissiveColor	Color [0..1]	Specifies the color of the light emitted by the surface geometry object.
isSmooth	Boolean [0..1]	Specifies which interpolation method is used for the shading of the surface geometry object. If the attribute is set to true, vertex normals should be used for shading (Gouraud shading). Otherwise, normals should be constant for a surface patch (flat shading).
shininess	DoubleBetween0and1 [0..1]	Specifies the sharpness of the specular highlight.
specularColor	Color [0..1]	Specifies the color of the light directly reflected by the surface geometry object.
target	URI [0..*]	Specifies the URI that points to the surface geometry objects to which the material is applied.
transparency	DoubleBetween0and1 [0..1]	Specifies the degree of transparency of the surface geometry object.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.5.10. Class AbstractTextureParameterization

AbstractTextureParameterization	
Definition:	AbstractTextureParameterization is the abstract superclass for different kinds of texture parameterizations.
Subclass Of:	None
StereoType:	«DataType»

9.1.5.11. Class TexCoordGen

TexCoordGen

Definition:	TexCoordGen defines texture parameterization using a transformation matrix.
Subclass Of:	None
StereoType:	«DataType»

Role name	Target class and multiplicity	Definition
crs	SC_CRS [0..1]	

Attribute	Value type and multiplicity	Definition
worldToTexture	TransformationMatrix3x4	Specifies the 3x4 transformation matrix that defines the transformation between world coordinates and texture coordinates.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.5.12. Class TexCoordList

TexCoordList

Definition:	TexCoordList defines texture parameterization using texture coordinates.
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
ring	URI [1..*]	Specifies the URIs that point to the LinearRings that are parameterized using the given texture coordinates.
textureCoordinates	DoubleList [1..*]	Specifies the coordinates of texture used for parameterization. The texture coordinates are provided separately for each LinearRing of the surface geometry object.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.5.13. Class TextureType

TextureType

Definition:	TextureType enumerates the different texture types.
Subclass Of:	None
StereoType:	

Attribute	Value type and multiplicity	Definition
specific		Indicates that the texture is specific to a single surface.
typical		Indicates that the texture is characteristic of a surface and can be used repeatedly.
unknown		Indicates that the texture type is not known.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.5.14. Class WrapMode

WrapMode

Definition:	WrapMode enumerates the different fill modes for textures.
Subclass Of:	None
StereoType:	

Attribute	Value type and multiplicity	Definition
none		Indicates that the texture is applied to the surface "as is". The part of the surface that is not covered by the texture is shown fully transparent. [cf. COLLADA]
wrap		Indicates that the texture is repeated until the surface is fully covered. [cf. COLLADA]
mirror		Indicates that the texture is repeated and mirrored. [cf. COLLADA]
clamp		Indicates that the texture is stretched to the edges of the surface. [cf. COLLADA]
border		Indicates that the texture is applied to the surface "as is". The part of the surface that is not covered by the texture is filled with the RGBA color that is specified in the attribute borderColor. [cf. COLLADA]

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.6. Package

Package Bridge	
Description:	The Bridge module supports representation of thematic and spatial aspects of bridges, bridge parts, bridge installations, and interior bridge structures.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.7. Classes

9.1.7.1. Class AbstractBridge

AbstractBridge	
Definition:	AbstractBridge is an abstract superclass representing the common attributes and associations of the classes Bridge and BridgePart.
Subclass Of:	AbstractConstruction
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
bridgeRoom	BridgeRoom [*]	
address	Address [*]	
bridgeConstructiveElement	BridgeConstructiveElement [*]	
bridgeInstallation	BridgeInstallation [*]	
bridgeFurniture	BridgeFurniture [*]	

Attribute	Value type and multiplicity	Definition
class	BridgeClassValue [0..1]	Indicates the specific type of the Bridge or BridgePart.
function	BridgeFunctionValue [0..*]	Specifies the intended purposes of the Bridge or BridgePart.
isMovable	Boolean [0..1]	Indicates whether the Bridge or BridgePart can be moved to allow for watercraft to pass.
usage	BridgeUsageValue [0..*]	Specifies the actual uses of the Bridge or BridgePart.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.7.2. Class Bridge

Bridge

Definition:	A Bridge represents a structure that affords the passage of pedestrians, animals, vehicles, and service(s) above obstacles or between two points at a height above ground. [cf. ISO 6707-1]
Subclass Of:	AbstractBridge
StereoType:	«TopLevelFeatureType»

Role name	Target class and multiplicity	Definition
bridgePart	BridgePart [*]	

9.1.7.3. Class BridgeClassValue

BridgeClassValue

Definition:	BridgeClassValue is a code list used to further classify a Bridge.
Subclass Of:	None
StereoType:	«CodeList»

9.1.7.4. Class BridgeConstructiveElement

BridgeConstructiveElement

Definition:	A BridgeConstructiveElement is an element of a bridge which is essential from a structural point of view. Examples are pylons, anchorages, slabs, beams.
Subclass Of:	AbstractConstructiveElement
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	BridgeConstructiveElementClassValue [0..1]	Indicates the specific type of the BridgeConstructiveElement.
function	BridgeConstructiveElementFunctionValue [0..*]	Specifies the intended purposes of the BridgeConstructiveElement.
usage	BridgeConstructiveElementUsageValue [0..*]	Specifies the actual uses of the BridgeConstructiveElement.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.7.5. Class BridgeConstructiveElementClassValue

BridgeConstructiveElementClassValue

Definition:	BridgeConstructiveElementClassValue is a code list used to further classify a BridgeConstructiveElement.
Subclass Of:	None
StereoType:	«CodeList»

9.1.7.6. Class BridgeConstructiveElementFunctionValue

BridgeConstructiveElementFunctionValue

Definition:	BridgeConstructiveElementFunctionValue is a code list that enumerates the different purposes of a BridgeConstructiveElement.
Subclass Of:	None
StereoType:	«CodeList»

9.1.7.7. Class BridgeConstructiveElementUsageValue

BridgeConstructiveElementUsageValue	
Definition:	BridgeConstructiveElementUsageValue is a code list that enumerates the different uses of a BridgeConstructiveElement.
Subclass Of:	None>
StereoType:	«CodeList»

9.1.7.8. Class BridgeFunctionValue

BridgeFunctionValue	
Definition:	BridgeFunctionValue is a code list that enumerates the different purposes of a Bridge.
Subclass Of:	None
StereoType:	«CodeList»

9.1.7.9. Class BridgeFurniture

BridgeFurniture	
Definition:	A BridgeFurniture is an equipment for occupant use, usually not fixed to the bridge. [cf. ISO 6707-1]
Subclass Of:	AbstractFurniture
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	BridgeFurnitureClassValue [0..1]	Indicates the specific type of the BridgeFurniture.
function	BridgeFurnitureFunctionValue [0..*]	Specifies the intended purposes of the BridgeFurniture.
usage	BridgeFurnitureUsageValue [0..*]	Specifies the actual uses of the BridgeFurniture.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.7.10. Class BridgeFurnitureClassValue

BridgeFurnitureClassValue	
Definition:	BridgeFurnitureClassValue is a code list used to further classify a BridgeFurniture.
Subclass Of:	None
StereoType:	«CodeList»

9.1.7.11. Class BridgeFurnitureFunctionValue

BridgeFurnitureFunctionValue	
Definition:	BridgeFurnitureFunctionValue is a code list that enumerates the different purposes of a BridgeFurniture.
Subclass Of:	None
StereoType:	«CodeList»

9.1.7.12. Class BridgeFurnitureUsageValue

BridgeFurnitureUsageValue	
Definition:	BridgeFurnitureUsageValue is a code list that enumerates the different uses of a BridgeFurniture.
Subclass Of:	None
StereoType:	«CodeList»

9.1.7.13. Class BridgeInstallation

BridgeInstallation	
Definition:	A BridgeInstallation is a permanent part of a Bridge (inside and/or outside) which does not have the significance of a BridgePart. In contrast to BridgeConstructiveElements, a BridgeInstallation is not essential from a structural point of view. Examples are stairs, antennas or railways.
Subclass Of:	AbstractInstallation
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	BridgeInstallationClassValue [0..1]	Indicates the specific type of the BridgeInstallation.
function	BridgeInstallationFunctionValue [0..*]	Specifies the intended purposes of the BridgeInstallation.
usage	BridgeInstallationUsageValue [0..*]	Specifies the actual uses of the BridgeInstallation.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.7.14. Class BridgeInstallationClassValue

BridgeInstallationClassValue	
Definition:	BridgeInstallationClassValue is a code list used to further classify a BridgeInstallation.
Subclass Of:	None
StereoType:	«CodeList»

9.1.7.15. Class BridgeInstallationFunctionValue

BridgeInstallationFunctionValue	
Definition:	BridgeInstallationFunctionValue is a code list that enumerates the different purposes of a BridgeInstallation.
Subclass Of:	None
StereoType:	«CodeList»

9.1.7.16. Class BridgeInstallationUsageValue

BridgeInstallationUsageValue	
Definition:	BridgeInstallationUsageValue is a code list that enumerates the different uses of a BridgeInstallation.
Subclass Of:	None
StereoType:	«CodeList»

9.1.7.17. Class BridgePart

BridgePart	
Definition:	A BridgePart is a physical or functional subdivision of a Bridge. It would be considered a Bridge, if it were not part of a collection of other BridgeParts.
Subclass Of:	AbstractBridge
StereoType:	«FeatureType»

9.1.7.18. Class BridgeRoom

BridgeRoom	
Definition:	A BridgeRoom is a space within a Bridge or BridgePart intended for human occupancy (e.g. a place of work or recreation) and/or containment (storage) of animals or things. A BridgeRoom is bounded physically and/or virtually (e.g. by ClosureSurfaces or GenericSurfaces).
Subclass Of:	AbstractUnoccupiedSpace
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
bridgeInstallation	BridgeInstallation [*]	
boundary	AbstractThematicSurface [*]	
bridgeFurniture	BridgeFurniture [*]	

Attribute	Value type and multiplicity	Definition
class	BridgeRoomClassValue [0..1]	Indicates the specific type of the BridgeRoom.
function	BridgeRoomFunctionValue [0..*]	Specifies the intended purposes of the BridgeRoom.
usage	BridgeRoomUsageValue [0..*]	Specifies the actual uses of the BridgeRoom.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.7.19. Class BridgeRoomClassValue

BridgeRoomClassValue	
Definition:	BridgeRoomClassValue is a code list used to further classify a BridgeRoom.
Subclass Of:	None
StereoType:	«CodeList»

9.1.7.20. Class BridgeRoomFunctionValue

BridgeRoomFunctionValue	
Definition:	BridgeRoomFunctionValue is a code list that enumerates the different purposes of a BridgeRoom.
Subclass Of:	None
StereoType:	«CodeList»

9.1.7.21. Class BridgeRoomUsageValue

BridgeRoomUsageValue	
Definition:	BridgeRoomUsageValue is a code list that enumerates the different uses of a BridgeRoom.
Subclass Of:	None
StereoType:	«CodeList»

9.1.7.22. Class BridgeUsageValue

BridgeUsageValue	
Definition:	BridgeUsageValue is a code list that enumerates the different uses of a Bridge.
Subclass Of:	None
StereoType:	«CodeList»

9.1.8. Package

Package Building

Description:	The Building module supports representation of thematic and spatial aspects of buildings, building parts, building installations, building subdivisions, and interior building structures.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.9. Classes

9.1.9.1. Class AbstractBuilding

AbstractBuilding

Definition:	AbstractBuilding is an abstract superclass representing the common attributes and associations of the classes Building and BuildingPart.
Subclass Of:	AbstractConstruction
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
buildingConstructiveElement	BuildingConstructiveElement [*]	
address	Address [*]	
buildingSubdivision	AbstractBuildingSubdivision [*]	
buildingFurniture	BuildingFurniture [*]	
buildingRoom	BuildingRoom [*]	
buildingInstallation	BuildingInstallation [*]	

Attribute	Value type and multiplicity	Definition
class	BuildingClassValue [0..1]	Indicates the specific type of the Building or BuildingPart.
function	BuildingFunctionValue [0..*]	Specifies the intended purposes of the Building or BuildingPart.
roofType	RoofTypeValue [0..1]	Indicates the shape of the roof of the Building or BuildingPart.
storeyHeightsAboveGround	MeasureOrNilReasonList [0..1]	Lists the heights of each storey above ground. The first value in the list denotes the height of the storey closest to the ground level, the last value denotes the height furthest away.
storeyHeightsBelowGround	MeasureOrNilReasonList [0..1]	Lists the height of each storey below ground. The first value in the list denotes the height of the storey closest to the ground level, the last value denotes the height furthest away.
storeysAboveGround	Integer [0..1]	Indicates the number of storeys positioned above ground level.
storeysBelowGround	Integer [0..1]	Indicates the number of storeys positioned below ground level.
usage	BuildingUsageValue [0..*]	Specifies the actual uses of the Building or BuildingPart.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.9.2. Class AbstractBuildingSubdivision

AbstractBuildingSubdivision

Definition:	AbstractBuildingSubdivision is the abstract superclass for different kinds of logical building subdivisions.
Subclass Of:	AbstractLogicalSpace
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
buildingInstallation	BuildingInstallation [*]	
buildingFurniture	BuildingFurniture [*]	
buildingRoom	BuildingRoom [*]	
buildingConstructiveElement	BuildingConstructiveElement [*]	

Attribute	Value type and multiplicity	Definition
class	BuildingSubdivisionClassValue [0..1]	Indicates the specific type of the building subdivision.
elevation	Elevation [0..*]	Specifies qualified elevations of the building subdivision in relation to a well-defined surface which is commonly taken as origin (e.g. geoid or water level). [cf. INSPIRE]
function	BuildingSubdivisionFunctionValue [0..*]	Specifies the intended purposes of the building subdivision.
sortKey	Real [0..1]	Defines an order among the objects that belong to the building subdivision. An example is the sorting of storeys.
usage	BuildingSubdivisionUsageValue [0..*]	Specifies the actual uses of the building subdivision.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.9.3. Class Building

Building	
Definition:	A Building is a free-standing, self-supporting construction that is roofed, usually walled, and can be entered by humans and is normally designed to stand permanently in one place. It is intended for human occupancy (e.g. a place of work or recreation), habitation and/or shelter of humans, animals or things.
Subclass Of:	AbstractBuilding
StereoType:	«TopLevelFeatureType»

Role name	Target class and multiplicity	Definition
buildingPart	BuildingPart [*]	

9.1.9.4. Class BuildingClassValue

BuildingClassValue

Definition:	BuildingClassValue is a code list used to further classify a Building.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.5. Class BuildingConstructiveElement

BuildingConstructiveElement

Definition:	A BuildingConstructiveElement is an element of a Building which is essential from a structural point of view. Examples are walls, slabs, staircases, beams.
Subclass Of:	AbstractConstructiveElement
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	BuildingConstructiveElementClassValue [0..1]	Indicates the specific type of the BuildingConstructiveElement.
function	BuildingConstructiveElementFunctionValue [0..*]	Specifies the intended purposes of the BuildingConstructiveElement.
usage	BuildingConstructiveElementUsageValue [0..*]	Specifies the actual uses of the BuildingConstructiveElement.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.9.6. Class BuildingConstructiveElementClassValue

BuildingConstructiveElementClassValue

Definition:	BuildingConstructiveElementClassValue is a code list used to further classify a BuildingConstructiveElement.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.7. Class BuildingConstructiveElementFunctionValue

BuildingConstructiveElementFunctionValue	
Definition:	BuildingConstructiveElementFunctionValue is a code list that enumerates the different purposes of a BuildingConstructiveElement.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.8. Class BuildingConstructiveElementUsageValue

BuildingConstructiveElementUsageValue	
Definition:	BuildingConstructiveElementUsageValue is a code list that enumerates the different uses of a BuildingConstructiveElement.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.9. Class BuildingFunctionValue

BuildingFunctionValue	
Definition:	BuildingFunctionValue is a code list that enumerates the different purposes of a Building.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.10. Class BuildingFurniture

BuildingFurniture

Definition:	A BuildingFurniture is an equipment for occupant use, usually not fixed to the building. [cf. ISO 6707-1]
Subclass Of:	AbstractFurniture
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	BuildingFurnitureClassValue [0..1]	Indicates the specific type of the BuildingFurniture.
function	BuildingFurnitureFunctionValue [0..*]	Specifies the intended purposes of the BuildingFurniture.
usage	BuildingFurnitureUsageValue [0..*]	Specifies the actual uses of the BuildingFurniture.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.9.11. Class BuildingFurnitureClassValue

BuildingFurnitureClassValue

Definition:	BuildingFurnitureClassValue is a code list used to further classify a BuildingFurniture.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.12. Class BuildingFurnitureFunctionValue

BuildingFurnitureFunctionValue

Definition:	BuildingFurnitureFunctionValue is a code list that enumerates the different purposes of a BuildingFurniture.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.13. Class BuildingFurnitureUsageValue

BuildingFurnitureUsageValue

Definition:	BuildingFurnitureUsageValue is a code list that enumerates the different uses of a BuildingFurniture.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.14. Class BuildingInstallation

BuildingInstallation

Definition:	A BuildingInstallation is a permanent part of a Building (inside and/or outside) which has not the significance of a BuildingPart. Examples are stairs, antennas, balconies or small roofs.
Subclass Of:	AbstractInstallation
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	BuildingInstallationClassValue [0..1]	Indicates the specific type of the BuildingInstallation.
function	BuildingInstallationFunctionValue [0..*]	Specifies the intended purposes of the BuildingInstallation.
usage	BuildingInstallationUsageValue [0..*]	Specifies the actual uses of the BuildingInstallation.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.9.15. Class BuildingInstallationClassValue

BuildingInstallationClassValue

Definition:	BuildingInstallationClassValue is a code list used to further classify a BuildingInstallation.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.16. Class BuildingInstallationFunctionValue

BuildingInstallationFunctionValue

Definition:	BuildingInstallationFunctionValue is a code list that enumerates the different purposes of a BuildingInstallation.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.17. Class BuildingInstallationUsageValue

BuildingInstallationUsageValue	
Definition:	BuildingInstallationUsageValue is a code list that enumerates the different uses of a BuildingInstallation.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.18. Class BuildingPart

BuildingPart	
Definition:	A BuildingPart is a physical or functional subdivision of a Building. It would be considered a Building, if it were not part of a collection of other BuildingParts.
Subclass Of:	AbstractBuilding
StereoType:	«FeatureType»

9.1.9.19. Class BuildingRoom

BuildingRoom	
Definition:	A BuildingRoom is a space within a Building or BuildingPart intended for human occupancy (e.g. a place of work or recreation) and/or containment of animals or things. A BuildingRoom is bounded physically and/or virtually (e.g. by ClosureSurfaces or GenericSurfaces).
Subclass Of:	AbstractUnoccupiedSpace
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
buildingFurniture	BuildingFurniture [*]	
boundary	AbstractThematicSurface [*]	
buildingInstallation	BuildingInstallation [*]	

Attribute	Value type and multiplicity	Definition
class	BuildingRoomClassValue [0..1]	Indicates the specific type of the BuildingRoom.
function	BuildingRoomFunctionValue [0..*]	Specifies the intended purposes of the BuildingRoom.
roomHeight	RoomHeight [0..*]	Specifies qualified heights of the BuildingRoom.
usage	BuildingRoomUsageValue [0..*]	Specifies the actual uses of the BuildingRoom.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.9.20. Class BuildingRoomClassValue

BuildingRoomClassValue	
Definition:	BuildingRoomClassValue is a code list used to further classify a BuildingRoom.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.21. Class BuildingRoomFunctionValue

BuildingRoomFunctionValue	
Definition:	BuildingRoomFunctionValue is a code list that enumerates the different purposes of a BuildingRoom.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.22. Class BuildingRoomUsageValue

BuildingRoomUsageValue	
Definition:	BuildingRoomUsageValue is a code list that enumerates the different uses of a BuildingRoom.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.23. Class BuildingSubdivisionClassValue

BuildingSubdivisionClassValue	
Definition:	BuildingSubdivisionClassValue is a code list used to further classify a BuildingSubdivision.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.24. Class BuildingSubdivisionFunctionValue

BuildingSubdivisionFunctionValue	
Definition:	BuildingSubdivisionFunctionValue is a code list that enumerates the different purposes of a BuildingSubdivision.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.25. Class BuildingSubdivisionUsageValue

BuildingSubdivisionUsageValue	
Definition:	BuildingSubdivisionUsageValue is a code list that enumerates the different uses of a BuildingSubdivision.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.26. Class BuildingUnit

BuildingUnit	
Definition:	A BuildingUnit is a logical subdivision of a Building. BuildingUnits are formed according to some homogeneous property like function, ownership, management, or accessability. They may be separately sold, rented out, inherited, managed, etc.
Subclass Of:	AbstractBuildingSubdivision
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
address	Address [*]	

9.1.9.27. Class BuildingUsageValue

BuildingUsageValue	
Definition:	BuildingUsageValue is a code list that enumerates the different uses of a Building.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.28. Class RoofTypeValue

RoofTypeValue	
Definition:	RoofTypeValue is a code list that enumerates different roof types.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.29. Class RoomElevationReferenceValue

RoomElevationReferenceValue	
Definition:	RoomElevationReferenceValue is a code list that enumerates the different elevation reference levels used to measure room heights.
Subclass Of:	None
StereoType:	«CodeList»

9.1.9.30. Class Storey

Storey	
Definition:	A Storey is a horizontal section of a Building.
Subclass Of:	AbstractBuildingSubdivision
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
buildingUnit	BuildingUnit [*]	
boundary	AbstractThematicSurface [*]	

9.1.9.31. Class RoomHeight

RoomHeight	
Definition:	The RoomHeight represents a vertical distance (measured or estimated) between a low reference and a high reference. [cf. INSPIRE]
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
highReference	RoomElevationReferenceValue	Indicates the high point used to calculate the value of the room height.
lowReference	RoomElevationReferenceValue	Indicates the low point used to calculate the value of the room height.
status	HeightStatusValue	Indicates the way the room height has been captured.
value	Length	Specifies the value of the room height.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.10. Package

Package CityFurniture

Description:	The CityFurniture module supports representation of city furniture objects. 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.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.11. Classes

9.1.11.1. Class CityFurniture

CityFurniture

Definition:	CityFurniture is an object or piece of equipment installed in the outdoor environment for various purposes. Examples include street signs, traffic signals, street lamps, benches, fountains.
Subclass Of:	AbstractOccupiedSpace
StereoType:	«TopLevelFeatureType»
Constraint:	sdf (OCL): inv: function='sdf'

Attribute	Value type and multiplicity	Definition
class	CityFurnitureClassValue [0..1]	Indicates the specific type of the CityFurniture.
function	CityFurnitureFunctionValue [0..*]	Specifies the intended purposes of the CityFurniture.
usage	CityFurnitureUsageValue [0..*]	Specifies the actual uses of the CityFurniture.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.11.2. Class CityFurnitureClassValue

CityFurnitureClassValue

Definition:	CityFurnitureClassValue is a code list used to further classify a CityFurniture.
Subclass Of:	None
StereoType:	«CodeList»

9.1.11.3. Class CityFurnitureFunctionValue

CityFurnitureFunctionValue	
Definition:	CityFurnitureFunctionValue is a code list that enumerates the different purposes of a CityFurniture.
Subclass Of:	None
StereoType:	«CodeList»

9.1.11.4. Class CityFurnitureUsageValue

CityFurnitureUsageValue	
Definition:	CityFurnitureUsageValue is a code list that enumerates the different uses of a CityFurniture.
Subclass Of:	None
StereoType:	«CodeList»

9.1.12. Package

Package CityObjectGroup	
Description:	The CityObjectGroup module supports grouping of city objects. Arbitrary city objects may be aggregated in groups according to user-defined criteria. A group may be further classified by application-specific attributes.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.13. Classes

9.1.13.1. Class CityObjectGroup

CityObjectGroup

Definition:	A CityObjectGroup represents an application-specific aggregation of city objects 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 member of a group may be qualified by a role name, reflecting the role each city object plays in the context of the group.
Subclass Of:	AbstractLogicalSpace
StereoType:	«TopLevelFeatureType»

Role name	Target class and multiplicity	Definition
parent	AbstractCityObject [0..1]	
groupMember	AbstractCityObject [*]	

Attribute	Value type and multiplicity	Definition
class	CityObjectGroupClassValue [0..1]	Indicates the specific type of the CityObjectGroup.
function	CityObjectGroupFunctionValue [0..*]	Specifies the intended purposes of the CityObjectGroup.
usage	CityObjectGroupUsageValue [0..*]	Specifies the actual usages of the CityObjectGroup.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.13.2. Class CityObjectGroupClassValue

CityObjectGroupClassValue

Definition:	CityObjectGroupClassValue is a code list used to further classify a CityObjectGroup.
Subclass Of:	None
StereoType:	«CodeList»

9.1.13.3. Class CityObjectGroupFunctionValue

CityObjectGroupFunctionValue

Definition:	CityObjectGroupFunctionValue is a code list that enumerates the different purposes of a CityObjectGroup.
Subclass Of:	None
StereoType:	«CodeList»

9.1.13.4. Class CityObjectGroupUsageValue

CityObjectGroupUsageValue

Definition:	CityObjectGroupUsageValue is a code list that enumerates the different uses of a CityObjectGroup.
Subclass Of:	None
StereoType:	«CodeList»

9.1.13.5. Class Role

Role

Definition:	Role qualifies the function of a city object within the CityObjectGroup.
Subclass Of:	None
StereoType:	«ObjectType»

Attribute	Value type and multiplicity	Definition
role	CharacterString	Describes the role the city object plays within the CityObjectGroup.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.14. Package

Package Construction

Description:	The Construction module supports representation of key elements of different types of constructions. These key elements include construction surfaces (e.g floor and ceiling), windows and doors, constructive elements (e.g. beams and slabs), installations, and furniture.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.15. Classes

9.1.15.1. Class AbstractConstruction

AbstractConstruction	
Definition:	AbstractConstruction is the abstract superclass for objects that are manufactured by humans from construction materials, are connected to earth and are intended to be permanent. A connection with the ground also exists when the construction rests by its own weight on the ground or is moveable limited on stationary rails or if the construction is intended to be used mainly stationary.
Subclass Of:	AbstractOccupiedSpace
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
boundary	AbstractThematicSurface [*]	

Attribute	Value type and multiplicity	Definition
conditionOfConstruction	ConditionOfConstructionValue [0..1]	Indicates the life-cycle status of the construction. [cf. INSPIRE]
constructionEvent	ConstructionEvent [0..*]	Describes specific events in the life-time of the construction.
dateOfConstruction	Date [0..1]	Indicates the date at which the construction was completed.
dateOfDemolition	Date [0..1]	Indicates the date at which the construction was demolished.
elevation	Elevation [0..*]	Specifies qualified elevations of the construction in relation to a well-defined surface which is commonly taken as origin (e.g. geoid or water level). [cf. INSPIRE]
height	Height [0..*]	Specifies qualified heights of the construction above ground or below ground. [cf. INSPIRE]
occupancy	Occupancy [0..*]	Provides qualified information on the residency of persons, animals, or other moveable objects in the construction.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.15.2. Class AbstractConstructionSurface

AbstractConstructionSurface

Definition:	AbstractConstructionSurface is the abstract superclass for different kinds of surfaces that bound a construction.
Subclass Of:	AbstractThematicSurface
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
fillingSurface	AbstractFillingSurface [*]	

9.1.15.3. Class AbstractConstructiveElement

AbstractConstructiveElement

Definition:	AbstractConstructiveElement is the abstract superclass for the representation of volumetric elements of a construction. Examples are walls, beams, slabs.
Subclass Of:	AbstractOccupiedSpace
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
filling	AbstractFillingElement [*]	
boundary	AbstractThematicSurface [*]	

Attribute	Value type and multiplicity	Definition
isStructuralElement	Boolean [0..1]	Indicates whether the constructive element is essential from a structural point of view. A structural element cannot be omitted without collapsing of the construction. Examples are pylons and anchorages of bridges.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.15.4. Class AbstractFillingElement

AbstractFillingElement

Definition:	AbstractFillingElement is the abstract superclass for different kinds of elements that fill the openings of constructive elements.
Subclass Of:	AbstractOccupiedSpace
StereoType:	«FeatureType»

9.1.15.5. Class AbstractFillingSurface

AbstractFillingSurface

Definition:	AbstractFillingSurface is the abstract superclass for different kinds of surfaces that seal openings filled by filling elements.
Subclass Of:	AbstractThematicSurface
StereoType:	«FeatureType»

9.1.15.6. Class AbstractFurniture

AbstractFurniture

Definition:	AbstractFurniture is the abstract superclass for the representation of furniture objects of a construction.
Subclass Of:	AbstractOccupiedSpace
StereoType:	«FeatureType»

9.1.15.7. Class AbstractInstallation

AbstractInstallation

Definition:	AbstractInstallation is the abstract superclass for the representation of installation objects of a construction.
Subclass Of:	AbstractOccupiedSpace
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
boundary	AbstractThematicSurface [*]	

Attribute	Value type and multiplicity	Definition
relationToConstruction	RelationToConstruction [0..1]	Indicates whether the installation is located inside and/or outside of the construction.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.15.8. Class CeilingSurface

CeilingSurface

Definition:	A CeilingSurface is a surface that represents the interior ceiling of a construction. An example is the ceiling of a room.
Subclass Of:	AbstractConstructionSurface
StereoType:	«FeatureType»

9.1.15.9. Class Door

Door

Definition:	A Door is a construction for closing an opening intended primarily for access or egress or both. [cf. ISO 6707-1]
Subclass Of:	AbstractFillingElement
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
boundary	DoorSurface [*]	
address	Address [*]	

Attribute	Value type and multiplicity	Definition
class	DoorClassValue [0..1]	Indicates the specific type of the Door.
function	DoorFunctionValue [0..*]	Specifies the intended purposes of the Door.
usage	DoorUsageValue [0..*]	Specifies the actual uses of the Door.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.15.10. Class DoorClassValue

DoorClassValue	
Definition:	DoorClassValue is a code list used to further classify a Door.
Subclass Of:	None
StereoType:	«CodeList»

9.1.15.11. Class DoorFunctionValue

DoorFunctionValue	
Definition:	DoorFunctionValue is a code list that enumerates the different purposes of a Door.
Subclass Of:	None
StereoType:	«CodeList»

9.1.15.12. Class DoorSurface

DoorSurface	
Definition:	A DoorSurface is either a boundary surface of a Door feature or a surface that seals an opening filled by a door.
Subclass Of:	AbstractFillingSurface
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
address	Address [*]	

9.1.15.13. Class DoorUsageValue

DoorUsageValue	
Definition:	DoorUsageValue is a code list that enumerates the different uses of a Door.
Subclass Of:	None
StereoType:	«CodeList»

9.1.15.14. Class ElevationReferenceValue

ElevationReferenceValue	
Definition:	ElevationReferenceValue is a code list that enumerates the different elevation reference levels used to measure construction heights.
Subclass Of:	None
StereoType:	«CodeList»

9.1.15.15. Class EventValue

EventValue	
Definition:	EventValue is a code list that enumerates the different events of a construction.
Subclass Of:	None
StereoType:	«CodeList»

9.1.15.16. Class FloorSurface

FloorSurface	
Definition:	A FloorSurface is surface that represents the interior floor of a construction. An example is the floor of a room.
Subclass Of:	AbstractConstructionSurface
StereoType:	«FeatureType»

9.1.15.17. Class GroundSurface

GroundSurface	
Definition:	A GroundSurface is a surface that represents the ground plate of a construction. The polygon defining the ground plate is congruent with the footprint of the construction.
Subclass Of:	AbstractConstructionSurface
StereoType:	«FeatureType»

9.1.15.18. Class InteriorWallSurface

InteriorWallSurface	
Definition:	An InteriorWallSurface is a surface that is visible from inside a construction. An example is the wall of a room.
Subclass Of:	AbstractConstructionSurface
StereoType:	«FeatureType»

9.1.15.19. Class OtherConstruction

OtherConstruction	
Definition:	An OtherConstruction is a construction that is not covered by any of the other subclasses of AbstractConstruction.
Subclass Of:	AbstractConstruction
StereoType:	«TopLevelFeatureType»

Attribute	Value type and multiplicity	Definition
class	OtherConstructionClassValue [0..1]	Indicates the specific type of the OtherConstruction.
function	OtherConstructionFunctionValue [0..*]	Specifies the intended purposes of the OtherConstruction.
usage	OtherConstructionUsageValue [0..*]	Specifies the actual uses of the OtherConstruction.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.15.20. Class OtherConstructionClassValue

OtherConstructionClassValue	
Definition:	OtherConstructionClassValue is a code list used to further classify an OtherConstruction.
Subclass Of:	None
StereoType:	«CodeList»

9.1.15.21. Class OtherConstructionFunctionValue

OtherConstructionFunctionValue	
Definition:	OtherConstructionFunctionValue is a code list that enumerates the different purposes of an OtherConstruction.
Subclass Of:	None
StereoType:	«CodeList»

9.1.15.22. Class OtherConstructionUsageValue

OtherConstructionUsageValue	
Definition:	OtherConstructionUsageValue is a code list that enumerates the different uses of an OtherConstruction.
Subclass Of:	None
StereoType:	«CodeList»

9.1.15.23. Class OuterCeilingSurface

OuterCeilingSurface	
Definition:	An OuterCeilingSurface is a surface that belongs to the outer building shell with the orientation pointing downwards. An example is the ceiling of a loggia.
Subclass Of:	AbstractConstructionSurface
StereoType:	«FeatureType»

9.1.15.24. Class OuterFloorSurface

OuterFloorSurface	
Definition:	An OuterFloorSurface is a surface that belongs to the outer construction shell with the orientation pointing upwards. An example is the floor of a loggia.
Subclass Of:	AbstractConstructionSurface
StereoType:	«FeatureType»

9.1.15.25. Class RoofSurface

RoofSurface	
Definition:	A RoofSurface is a surface that delimits major roof parts of a construction.
Subclass Of:	AbstractConstructionSurface
StereoType:	«FeatureType»

9.1.15.26. Class WallSurface

WallSurface	
Definition:	A WallSurface is a surface that is part of the building facade visible from the outside.
Subclass Of:	AbstractConstructionSurface
StereoType:	«FeatureType»

9.1.15.27. Class Window

Window	
Definition:	A Window is a construction for closing an opening in a wall or roof, primarily intended to admit light and/or provide ventilation. [cf. ISO 6707-1]
Subclass Of:	AbstractFillingElement
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
boundary	WindowSurface [*]	

Attribute	Value type and multiplicity	Definition
class	WindowClassValue [0..1]	Indicates the specific type of the Window.
function	WindowFunctionValue [0..*]	Specifies the intended purposes of the Window.
usage	WindowUsageValue [0..*]	Specifies the actual uses of the Window.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.15.28. Class WindowClassValue

WindowClassValue	
Definition:	WindowClassValue is a code list used to further classify a Window.
Subclass Of:	None
StereoType:	«CodeList»

9.1.15.29. Class WindowFunctionValue

WindowFunctionValue	
Definition:	WindowFunctionValue is a code list that enumerates the different purposes of a Window.
Subclass Of:	None
StereoType:	«CodeList»

9.1.15.30. Class WindowSurface

WindowSurface	
Definition:	A WindowSurface is either a boundary surface of a Window feature or a surface that seals an opening filled by a window.
Subclass Of:	AbstractFillingSurface
StereoType:	«FeatureType»

9.1.15.31. Class WindowUsageValue

WindowUsageValue	
Definition:	WindowUsageValue is a code list that enumerates the different uses of a Window.
Subclass Of:	None
StereoType:	«CodeList»

9.1.15.32. Class ConditionOfConstructionValue

ConditionOfConstructionValue

Definition:	ConditionOfConstructionValue enumerates different conditions of a construction. [cf. INSPIRE]
Subclass Of:	None
StereoType:	

Attribute	Value type and multiplicity	Definition
declined		Indicates that the construction cannot be used under normal conditions, though its main elements (walls, roof) are still present. [cf. INSPIRE]
demolished		Indicates that the construction has been demolished. There are no more visible remains. [cf. INSPIRE]
functional		Indicates that the construction is functional. [cf. INSPIRE]
projected		Indicates that the construction is being designed. Construction works have not yet started. [cf. INSPIRE]
ruin		Indicates that the construction has been partly demolished and some main elements (roof, walls) have been destroyed. There are some visible remains of the construction. [cf. INSPIRE]
underConstruction		Indicates that the construction is under construction and not yet functional. This applies only to the initial construction works of the construction and not to maintenance work. [cf. INSPIRE]

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.15.33. Class ConstructionEvent

ConstructionEvent

Definition:	A ConstructionEvent is a data type used to describe a specific event that is associated with a construction. Examples are the issuing of a building permit or the renovation of a building.
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
dateOfEvent	Date	Specifies the date at which the event took or will take place.
description	CharacterString [0..1]	Provides additional information on the event.
event	EventValue	Indicates the specific event type.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.15.34. Class Elevation

Elevation

Definition:	Elevation is a data type that includes the elevation value itself and information on how this elevation was measured. [cf. INSPIRE]
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
elevationReference	ElevationReferenceValue	Specifies the level from which the elevation was measured. [cf. INSPIRE]
elevationValue	DirectPosition	Specifies the value of the elevation. [cf. INSPIRE]

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.15.35. Class Height

Height

Definition:	Height represents a vertical distance (measured or estimated) between a low reference and a high reference. [cf. INSPIRE]
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
highReference	ElevationReferenceValue	Indicates the high point used to calculate the value of the height. [cf. INSPIRE]
lowReference	ElevationReferenceValue	Indicates the low point used to calculate the value of the height. [cf. INSPIRE]
status	HeightStatusValue	Indicates the way the height has been captured. [cf. INSPIRE]
value	Length	Specifies the value of the height above or below ground. [cf. INSPIRE]

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.15.36. Class HeightStatusValue

HeightStatusValue

Definition:	HeightStatusValue enumerates the different methods used to capture a height. [cf. INSPIRE]
Subclass Of:	None
StereoType:	

Attribute	Value type and multiplicity	Definition
estimated		Indicates that the height has been estimated and not measured. [cf. INSPIRE]
measured		Indicates that the height has been (directly or indirectly) measured. [cf. INSPIRE]

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.15.37. Class RelationToConstruction

RelationToConstruction

Definition:	RelationToConstruction is an enumeration used to describe whether an installation is positioned inside and/or outside of a construction.
Subclass Of:	None
StereoType:	

Attribute	Value type and multiplicity	Definition
inside		Indicates that the installation is positioned inside of the construction.
outside		Indicates that the installation is positioned outside of the construction.
bothInsideAndOutside		Indicates that the installation is positioned inside as well as outside of the construction.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.16. Package

Package Dynamizer

Description:	The Dynamizer module supports the injection of timeseries data for individual attributes of CityGML features. Timeseries data can either be retrieved from external Sensor APIs (e.g. OGC SensorThings API, OGC Sensor Observation Services, MQTT, proprietary platforms), external standardized timeseries files (e.g. OGC TimeseriesML or OGC Observations & Measurements), external tabulated files (e.g CSV) or can be represented inline as basic time-value pairs.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.17. Classes

9.1.17.1. Class AbstractAtomicTimeseries

AbstractAtomicTimeseries

Definition:	AbstractAtomicTimeseries represents the attributes and relationships that are common to all kinds of atomic timeseries (GenericTimeseries, TabulatedFileTimeseries, StandardFileTimeseries). An atomic timeseries represents time-varying data of a specific data type for a single contiguous time interval.
Subclass Of:	AbstractTimeseries
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
observationProperty	CharacterString	Specifies the phenomenon for which the atomic timeseries provides observation values.
uom	CharacterString [0..1]	Specifies the unit of measurement of the observation values.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.17.2. Class AbstractTimeseries

AbstractTimeseries

Definition:	AbstractTimeseries is the abstract superclass representing any type of timeseries data.
Subclass Of:	AbstractFeature
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
firstTimestamp	TM_Position [0..1]	Specifies the beginning of the timeseries.
lastTimestamp	TM_Position [0..1]	Specifies the end of the timeseries.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.17.3. Class AuthenticationTypeValue

AuthenticationTypeValue

Definition:	AuthenticationTypeValue is a code list used to specify the authentication method to be used to access the referenced sensor service. Each value shall provide enough information such that a software application could determine the required access credentials.
Subclass Of:	None
StereoType:	«CodeList»

9.1.17.4. Class CompositeTimeseries

CompositeTimeseries

Definition:	A CompositeTimeseries is a (possibly recursive) aggregation of atomic and composite timeseries. The components of a composite timeseries must have non-overlapping time intervals.
Subclass Of:	AbstractTimeseries
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
component	TimeseriesComponent [1..*]	

9.1.17.5. Class Dynamizer

Dynamizer

Definition:	A Dynamizer is an object that injects timeseries data for an individual attribute of the city object in which it is included. The timeseries data overrides the static value of the referenced city object attribute in order to represent dynamic, i.e. time-dependent, variations of its value.
Subclass Of:	AbstractDynamizer
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
dynamicData	AbstractTimeseries [0..1]	
sensorConnection	SensorConnection [0..1]	

Attribute	Value type and multiplicity	Definition
attributeRef	CharacterString	Specifies the attribute of a CityGML feature whose value is overridden or replaced by the (dynamic) values specified by the Dynamizer.
endTime	TM_Position [0..1]	Specifies the end of the time span for which the Dynamizer provides dynamic values.
startTime	TM_Position [0..1]	Specifies the beginning of the time span for which the Dynamizer provides dynamic values.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.17.6. Class GenericTimeseries

GenericTimeseries	
Definition:	A GenericTimeseries represents time-varying data in the form of embedded time-value-pairs of a specific data type for a single contiguous time interval.
Subclass Of:	AbstractAtomicTimeseries
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
timeValuePair	TimeValuePair [1..*]	

Attribute	Value type and multiplicity	Definition
valueType	TimeseriesTypeValue	Indicates the specific type of all time-value-pairs that are part of the GenericTimeseries.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.17.7. Class SensorConnectionTypeValue

SensorConnectionTypeValue	
Definition:	SensorConnectionTypeValue is a code list used to specify the type of the referenced sensor service. Each value shall provide enough information such that a software application would be able to identify the API type and version.
Subclass Of:	None
StereoType:	«CodeList»

9.1.17.8. Class StandardFileTimeseries

StandardFileTimeseries

Definition:	A StandardFileTimeseries represents time-varying data for a single contiguous time interval. The data is provided in an external file referenced in the StandardFileTimeseries. The data within the external file shall be encoded according to a dedicated format for the representation of timeseries data, for example, the OGC TimeseriesML or OGC Observations & Measurements standard. The data type of the data has to be specified within the external file.
Subclass Of:	AbstractAtomicTimeseries
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
fileLocation	URI	Specifies the URI that points to the external timeseries file.
fileType	StandardFileTypeValue	Specifies the format used to represent the timeseries data.
mimeType	MimeTypeValue [0..1]	Specifies the MIME type of the external timeseries file.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.17.9. Class StandardFileTypeValue

StandardFileTypeValue

Definition:	StandardFileTypeValue is a code list used to specify the type of the referenced external timeseries data file. Each value shall provide information about the standard and version.
Subclass Of:	None
StereoType:	«CodeList»

9.1.17.10. Class TabulatedFileTimeseries

TabulatedFileTimeseries

Definition:	A TabulatedFileTimeseries represents time-varying data of a specific data type for a single contiguous time interval. The data is provided in an external file referenced in the TabulatedFileTimeseries. The file shall contain table structured data using an appropriate file format like comma separated values (CSV), Microsoft Excel (XLSX) or Google Spreadsheet. The timestamps and the values are given in specific columns of the table. Each row represents a single time-value-pair. A subset of rows can be selected using the idColumn and idValue attributes.
Subclass Of:	AbstractAtomicTimeseries
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
decimalSymbol	Character [0..1]	Indicates which symbol is used to separate the integer part from the fractional part of a decimal number.
fieldSeparator	CharacterString	Indicates which symbol is used to separate the individual values in the tabulated file.
fileLocation	URI	Specifies the URI that points to the external timeseries file.
fileType	TabulatedFileTypeValue	Specifies the format used to represent the timeseries data.
idColumnName	CharacterString [0..1]	Specifies the name of the column that stores the identifier of the time-value-pair.
idColumnNo	Integer [0..1]	Specifies the number of the column that stores the identifier of the time-value-pair.
idValue	CharacterString [0..1]	Specifies the value of the identifier for which the time-value-pairs are to be selected.
mimeType	MimeTypeValue [0..1]	Specifies the MIME type of the external timeseries file.
numberOfHeaderLines	Integer [0..1]	Indicates the number of lines at the beginning of the tabulated file that represent headers.
timeColumnName	CharacterString [0..1]	Specifies the name of the column that stores the timestamp of the time-value-pair.
timeColumnNo	Integer [0..1]	Specifies the number of the column that stores the timestamp of the time-value-pair.
valueColumnName	CharacterString [0..1]	Specifies the name of the column that stores the value of the time-value-pair.
valueColumnNo	Integer [0..1]	Specifies the number of the column that stores the value of the time-value-pair.
valueType	TimeseriesTypeValue	Indicates the specific type of the timeseries data.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.17.11. Class TabulatedFileTypeValue

TabulatedFileTypeValue

Definition:	TabulatedFileTypeValue is a code list used to specify the data format of the referenced external tabulated data file.
Subclass Of:	None
StereoType:	«CodeList»

9.1.17.12. Class SensorConnection

SensorConnection

Definition:	A SensorConnection provides all details that are required to retrieve a specific datastream from an external sensor web service. It comprises the service type (e.g. OGC SensorThings API, OGC Sensor Observation Services, MQTT, proprietary platforms), the URL of the sensor service, the identifier for the sensor or thing and its observed property as well as information about the required authentication method.
Subclass Of:	None
StereoType:	«DataType»

Role name	Target class and multiplicity	Definition
sensorLocation	AbstractCityObject [0..1]	

Attribute	Value type and multiplicity	Definition
authType	AuthenticationTypeValue [0..1]	Specifies the type of authentication required to be able to access the Sensor API.
baseURL	URI [0..1]	Specifies the base URL of the Sensor API request.
connectionType	SensorConnectionTypeValue	Indicates the type of Sensor API to which the SensorConnection refers.
datastreamID	CharacterString [0..1]	Specifies the datastream that is retrieved by the SensorConnection.
linkToObservation	CharacterString [0..1]	Specifies the complete URL to the observation request.
linkToSensorDescription	CharacterString [0..1]	Specifies the complete URL to the sensor description request.
mqttServer	CharacterString [0..1]	Specifies the name of the MQTT Server. This attribute is relevant when the MQTT Protocol is used to connect to a Sensor API.
mqttTopic	CharacterString [0..1]	Names the specific datastream that is retrieved by the SensorConnection.
observationID	CharacterString [0..1]	Specifies the unique identifier of the observation that is retrieved by the SensorConnection.
observationProperty	CharacterString	Specifies the phenomenon for which the SensorConnection provides observations.
sensorID	CharacterString [0..1]	Specifies the unique identifier of the sensor from which the SensorConnection retrieves observations.
sensorName	CharacterString [0..1]	Specifies the name of the sensor from which the SensorConnection retrieves observations.
uom	CharacterString [0..1]	Specifies the unit of measurement of the observations.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.17.13. Class TimeseriesComponent

TimeseriesComponent	
Definition:	TimeseriesComponent represents an element of a CompositeTimeseries.
Subclass Of:	None
StereoType:	«DataType»

Role name	Target class and multiplicity	Definition
timeseries	AbstractTimeseries [1]	

Attribute	Value type and multiplicity	Definition
additionalGap	TM_Duration [0..1]	Specifies how much extra time is added after all repetitions as an additional gap.
repetitions	Integer	Specifies how often the timeseries that is referenced by the TimeseriesComponent should be iterated.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.17.14. Class TimeseriesTypeValue

TimeseriesTypeValue

Definition:	TimeseriesTypeValue enumerates the possible value types for GenericTimeseries and TimeValuePair.
Subclass Of:	None
StereoType:	

Attribute	Value type and multiplicity	Definition
integer		Indicates that the values of the GenericTimeseries are of type "Integer".
double		Indicates that the values of the GenericTimeseries are of type "Double".
string		Indicates that the values of the GenericTimeseries are of type "String".
geometry		Indicates that the values of the GenericTimeseries are geometries.
uri		Indicates that the values of the GenericTimeseries are of type "URI".
bool		Indicates that the values of the GenericTimeseries are of type "Boolean".
implicitGeometry		Indicates that the values of the GenericTimeseries are of type "ImplicitGeometry".
appearance		Indicates that the values of the GenericTimeseries are of type "Appearance".

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.17.15. Class TimeValuePair

TimeValuePair	
Definition:	A TimeValuePair represents a value that is valid for a given timepoint. For each TimeValuePair only one of the value properties can be used mutually exclusive. Which value property has to be provided depends on the selected value type in the GenericTimeSeries feature, in which the TimeValuePair is included.
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
appearanceValue	AbstractAppearance [0..1]	Specifies the "Appearance" value of the TimeValuePair.
boolValue	Boolean [0..1]	Specifies the "Boolean" value of the TimeValuePair.
doubleValue	Real [0..1]	Specifies the "Double" value of the TimeValuePair.
geometryValue	GM_Object [0..1]	Specifies the geometry value of the TimeValuePair.
implicitGeometryValue	ImplicitGeometry [0..1]	Specifies the "ImplicitGeometry" value of the TimeValuePair.
intValue	Integer [0..1]	Specifies the "Integer" value of the TimeValuePair.
stringValue	CharacterString [0..1]	Specifies the "String" value of the TimeValuePair.
timestamp	TM_Position	Specifies the timepoint at which the value of the TimeValuePair is valid.
uriValue	URI [0..1]	Specifies the "URI" value of the TimeValuePair.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.18. Package

Package Generics

Description:	The Generics module supports application-specific extensions to the CityGML data model. These extensions may be used to model and exchange additional attributes and features not covered by the predefined thematic classes of CityGML. Generic extensions shall only be used if appropriate thematic classes or attributes are not provided by any other CityGML module.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.19. Classes

9.1.19.1. Class GenericLogicalSpace

GenericLogicalSpace

Definition:	A GenericLogicalSpace is a space that is not represented by any explicitly modelled AbstractLogicalSpace subclass within CityGML.
Subclass Of:	AbstractLogicalSpace
StereoType:	«TopLevelFeatureType»

Attribute	Value type and multiplicity	Definition
class	GenericLogicalSpaceClassValue [0..1]	Indicates the specific type of the GenericLogicalSpace.
function	GenericLogicalSpaceFunctionValue [0..*]	Specifies the intended purposes of the GenericLogicalSpace.
usage	GenericLogicalSpaceUsageValue [0..*]	Specifies the actual uses of the GenericLogicalSpace.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.19.2. Class GenericLogicalSpaceClassValue

GenericLogicalSpaceClassValue

Definition:	GenericLogicalSpaceClassValue is a code list used to further classify a GenericLogicalSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.19.3. Class GenericLogicalSpaceFunctionValue

GenericLogicalSpaceFunctionValue

Definition:	GenericLogicalSpaceFunctionValue is a code list that enumerates the different purposes of a GenericLogicalSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.19.4. Class GenericLogicalSpaceUsageValue

GenericLogicalSpaceUsageValue

Definition:	GenericLogicalSpaceUsageValue is a code list that enumerates the different uses of a GenericLogicalSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.19.5. Class GenericOccupiedSpace

GenericOccupiedSpace

Definition:	A GenericOccupiedSpace is a space that is not represented by any explicitly modelled AbstractOccupiedSpace subclass within CityGML.
Subclass Of:	AbstractOccupiedSpace
StereoType:	«TopLevelFeatureType»

Attribute	Value type and multiplicity	Definition
class	GenericOccupiedSpaceClassValue [0..1]	Indicates the specific type of the GenericOccupiedSpace.
function	GenericOccupiedSpaceFunctionValue [0..*]	Specifies the intended purposes of the GenericOccupiedSpace.
usage	GenericOccupiedSpaceUsageValue [0..*]	Specifies the actual uses of the GenericOccupiedSpace.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.19.6. Class GenericOccupiedSpaceClassValue

GenericOccupiedSpaceClassValue

Definition:	GenericOccupiedSpaceClassValue is a code list used to further classify a GenericOccupiedSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.19.7. Class GenericOccupiedSpaceFunctionValue

GenericOccupiedSpaceFunctionValue

Definition:	GenericOccupiedSpaceFunctionValue is a code list that enumerates the different purposes of a GenericOccupiedSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.19.8. Class GenericOccupiedSpaceUsageValue

GenericOccupiedSpaceUsageValue

Definition:	GenericOccupiedSpaceUsageValue is a code list that enumerates the different uses of a GenericOccupiedSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.19.9. Class GenericThematicSurface

GenericThematicSurface

Definition:	A GenericThematicSurface is a surface that is not represented by any explicitly modelled AbstractThematicSurface subclass within CityGML.
Subclass Of:	AbstractThematicSurface
StereoType:	«TopLevelFeatureType»

Attribute	Value type and multiplicity	Definition
class	GenericThematicSurfaceClassValue [0..1]	Indicates the specific type of the GenericThematicSurface.
function	GenericThematicSurfaceFunctionValue [0..*]	Specifies the intended purposes of the GenericThematicSurface.
usage	GenericThematicSurfaceUsageValue [0..*]	Specifies the actual uses of the GenericThematicSurface.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.19.10. Class GenericThematicSurfaceClassValue

GenericThematicSurfaceClassValue	
Definition:	GenericThematicSurfaceClassValue is a code list used to further classify a GenericThematicSurface.
Subclass Of:	None
StereoType:	«CodeList»

9.1.19.11. Class GenericThematicSurfaceFunctionValue

GenericThematicSurfaceFunctionValue	
Definition:	GenericThematicSurfaceFunctionValue is a code list that enumerates the different purposes of a GenericThematicSurface.
Subclass Of:	None
StereoType:	«CodeList»

9.1.19.12. Class GenericThematicSurfaceUsageValue

GenericThematicSurfaceUsageValue	
Definition:	GenericThematicSurfaceUsageValue is a code list that enumerates the different uses of a GenericThematicSurface.
Subclass Of:	None
StereoType:	«CodeList»

9.1.19.13. Class GenericUnoccupiedSpace

GenericUnoccupiedSpace	
Definition:	A GenericUnoccupiedSpace is a space that is not represented by any explicitly modelled AbstractUnoccupiedSpace subclass within CityGML.
Subclass Of:	AbstractUnoccupiedSpace
StereoType:	«TopLevelFeatureType»

Attribute	Value type and multiplicity	Definition
class	GenericUnoccupiedSpaceClassValue [0..1]	Indicates the specific type of the GenericUnoccupiedSpace.
function	GenericUnoccupiedSpaceFunctionValue [0..*]	Specifies the intended purposes of the GenericUnoccupiedSpace.
usage	GenericUnoccupiedSpaceUsageValue [0..*]	Specifies the actual uses of the GenericUnoccupiedSpace.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.19.14. Class GenericUnoccupiedSpaceClassValue

GenericUnoccupiedSpaceClassValue	
Definition:	GenericUnoccupiedSpaceClassValue is a code list used to further classify a GenericUnoccupiedSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.19.15. Class GenericUnoccupiedSpaceFunctionValue

GenericUnoccupiedSpaceFunctionValue	
Definition:	GenericUnoccupiedSpaceFunctionValue is a code list that enumerates the different purposes of a GenericUnoccupiedSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.19.16. Class GenericUnoccupiedSpaceUsageValue

GenericUnoccupiedSpaceUsageValue

Definition:	GenericUnoccupiedSpaceUsageValue is a code list that enumerates the different uses of a GenericUnoccupiedSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.19.17. Class DateAttribute

DateAttribute

Definition:	DateAttribute is a data type used to define generic attributes of type "Date".
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
name	CharacterString	Specifies the name of the DateAttribute.
value	Date	Specifies the "Date" value.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.19.18. Class DoubleAttribute

DoubleAttribute

Definition:	DoubleAttribute is a data type used to define generic attributes of type "Double".
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
name	CharacterString	Specifies the name of the DoubleAttribute.
value	Real	Specifies the "Double" value.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.19.19. Class GenericAttributeSet

GenericAttributeSet

Definition:	A GenericAttributeSet is a named collection of generic attributes.
Subclass Of:	None
StereoType:	«DataType»

Role name	Target class and multiplicity	Definition
genericAttribute	AbstractGenericAttribute [1..*]	

Attribute	Value type and multiplicity	Definition
codeSpace	URI [0..1]	Associates the GenericAttributeSet with an authority that maintains the collection of generic attributes.
name	CharacterString	Specifies the name of the GenericAttributeSet.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.19.20. Class IntAttribute

IntAttribute

Definition:	IntAttribute is a data type used to define generic attributes of type "Integer".
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
name	CharacterString	Specifies the name of the IntAttribute.
value	Integer	Specifies the "Integer" value.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.19.21. Class MeasureAttribute

MeasureAttribute

Definition:	MeasureAttribute is a data type used to define generic attributes of type "Measure".
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
name	CharacterString	Specifies the name of the MeasureAttribute.
value	Measure	Specifies the value of the MeasureAttribute. The value is of type "Measure", which can additionally provide the units of measure. [cf. ISO 19103]

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.19.22. Class StringAttribute

StringAttribute

Definition:	StringAttribute is a data type used to define generic attributes of type "String".
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
name	CharacterString	Specifies the name of the StringAttribute.
value	CharacterString	Specifies the "String" value.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.19.23. Class UriAttribute

UriAttribute

Definition:	UriAttribute is a data type used to define generic attributes of type "URI".
Subclass Of:	None
StereoType:	«DataType»

Attribute	Value type and multiplicity	Definition
name	CharacterString	Specifies the name of the UriAttribute.
value	URI	Specifies the "URI" value.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.20. Package

Package LandUse

Description:	The LandUse module supports representation of areas of the earth's surface dedicated to a specific land use.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.21. Classes

9.1.21.1. Class LandUse

LandUse

Definition:	A LandUse object is an area of the earth's surface dedicated to a specific land use or having a specific land cover with or without vegetation, such as sand, rock, mud flats, forest, or grasslands.
Subclass Of:	AbstractThematicSurface
StereoType:	«TopLevelFeatureType»

Attribute	Value type and multiplicity	Definition
class	LandUseClassValue [0..1]	Indicates the specific type of the LandUse.
function	LandUseFunctionValue [0..*]	Specifies the intended purposes of the LandUse.
usage	LandUseUsageValue [0..*]	Specifies the actual uses of the LandUse.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.21.2. Class LandUseClassValue

LandUseClassValue	
Definition:	LandUseClassValue is a code list used to further classify a LandUse.
Subclass Of:	None
StereoType:	«CodeList»

9.1.21.3. Class LandUseFunctionValue

LandUseFunctionValue	
Definition:	LandUseFunctionValue is a code list that enumerates the different purposes of a LandUse.
Subclass Of:	None
StereoType:	«CodeList»

9.1.21.4. Class LandUseUsageValue

LandUseUsageValue	
Definition:	LandUseUsageValue is a code list that enumerates the different uses of a LandUse.
Subclass Of:	None
StereoType:	«CodeList»

9.1.22. Package

Package PointCloud	
Description:	The PointCloud module supports representation of CityGML features by a collection of points.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.23. Classes

9.1.23.1. Class PointCloud

PointCloud	
Definition:	A PointCloud is an unordered collection of points that is a sampling of the geometry of a space or space boundary.
Subclass Of:	AbstractPointCloud
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
points	GM_MultiPoint [0..1]	

Attribute	Value type and multiplicity	Definition
mimeType	MimeTypeValue [0..1]	Specifies the MIME type of the external point cloud file.
pointFile	URI [0..1]	Specifies the URI that points to the external point cloud file.
pointFileSrsName	CharacterString [0..1]	Indicates the coordinate reference system used by the external point cloud file.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.24. Package

Package Relief	
Description:	The Relief module supports representation of the terrain. CityGML supports terrain representations at different levels of detail, reflecting different accuracies or resolutions. Terrain may be specified as a regular raster or grid, as a TIN, by break lines, and/or by mass points.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.25. Classes

9.1.25.1. Class AbstractReliefComponent

AbstractReliefComponent

Definition:	An AbstractReliefComponent represents an element of the terrain surface - either a TIN, a Grid, mass points or break lines.
Subclass Of:	AbstractSpaceBoundary
StereoType:	«FeatureType»
Constraint:	polygonGeometry (OCL): inv: self.extent.patch→size()=1 and self.extent.patch→forAll(oclIsKindOf(GM_Polygon))

Role name	Target class and multiplicity	Definition
extent	GM_Surface [0..1]	

Attribute	Value type and multiplicity	Definition
lod	IntegerBetween0and3	Indicates the Level of Detail of the terrain component.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.25.2. Class BreaklineRelief

BreaklineRelief

Definition:	A BreaklineRelief represents a terrain component with 3D lines. These lines denote break lines or ridge/valley lines.
Subclass Of:	AbstractReliefComponent
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
ridgeOrValleyLines	GM_MultiCurve [0..1]	
breaklines	GM_MultiCurve [0..1]	

9.1.25.3. Class MassPointRelief

MassPointRelief

Definition:	A MassPointRelief represents a terrain component as a collection of 3D points.
Subclass Of:	AbstractReliefComponent
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
reliefPoints	GM_MultiPoint [0..1]	
pointCloud	AbstractPointCloud [0..1]	

9.1.25.4. Class RasterRelief

RasterRelief

Definition:	A RasterRelief represents a terrain component as a regular raster or grid.
Subclass Of:	AbstractReliefComponent
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
grid	CV_DiscreteGridPointCoverage [1]	

9.1.25.5. Class ReliefFeature

ReliefFeature

Definition:	A ReliefFeature is a collection of terrain components representing the Earth's surface, a.k.a. the Digital Terrain Model.
Subclass Of:	AbstractSpaceBoundary
StereoType:	«TopLevelFeatureType»

Role name	Target class and multiplicity	Definition
reliefComponent	AbstractReliefComponent [1..*]	

Attribute	Value type and multiplicity	Definition
lod	IntegerBetween0and3	Indicates the Level of Detail of the ReliefFeature.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.25.6. Class TINRelief

TINRelief

Definition:	A TINRelief represents a terrain component as a triangulated irregular network.
Subclass Of:	AbstractReliefComponent
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
tin	GM_TriangulatedSurface [1]	

9.1.26. Package

Package Transportation

Description:	The Transportation module supports representation of the transportation infrastructure. Transportation features include roads, tracks, waterways, railways, and squares. Transportation features may be represented as a network and/or as a collection of spaces or surface elements embedded in a three-dimensional space.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.27. Classes

9.1.27.1. Class AbstractTransportationSpace

AbstractTransportationSpace

Definition:	AbstractTransportationSpace is the abstract superclass of transportation objects such as Roads, Tracks, Railways, Waterways or Squares.
Subclass Of:	AbstractUnoccupiedSpace
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
trafficSpace	TrafficSpace [*]	
hole	Hole [*]	
auxiliaryTrafficSpace	AuxiliaryTrafficSpace [*]	
marking	Marking [*]	

Attribute	Value type and multiplicity	Definition
occupancy	Occupancy [0..*]	Provides information on the residency of persons, vehicles, or other moving features in the transportation space.
trafficDirection	TrafficDirectionValue [0..1]	Indicates the direction of traffic flow relative to the corresponding linear geometry representation.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.2. Class AuxiliaryTrafficArea

AuxiliaryTrafficArea

Definition:	An AuxiliaryTrafficArea is the ground surface of an AuxiliaryTrafficSpace.
Subclass Of:	AbstractThematicSurface
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	AuxiliaryTrafficAreaClassValue [0..1]	Indicates the specific type of the AuxiliaryTrafficArea.
function	AuxiliaryTrafficAreaFunctionValue [0..*]	Specifies the intended purposes of the AuxiliaryTrafficArea.
surfaceMaterial	SurfaceMaterialValue [0..1]	Specifies the type of pavement of the AuxiliaryTrafficArea.
usage	AuxiliaryTrafficAreaUsageValue [0..*]	Specifies the actual uses of the AuxiliaryTrafficArea.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.3. Class AuxiliaryTrafficAreaClassValue

AuxiliaryTrafficAreaClassValue

Definition:	AuxiliaryTrafficAreaClassValue is a code list used to further classify an AuxiliaryTrafficArea.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.4. Class AuxiliaryTrafficAreaFunctionValue

AuxiliaryTrafficAreaFunctionValue

Definition:	AuxiliaryTrafficAreaFunctionValue is a code list that enumerates the different purposes of an AuxiliaryTrafficArea.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.5. Class AuxiliaryTrafficAreaUsageValue

AuxiliaryTrafficAreaUsageValue

Definition:	AuxiliaryTrafficAreaUsageValue is a code list that enumerates the different uses of an AuxiliaryTrafficArea.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.6. Class AuxiliaryTrafficSpace

AuxiliaryTrafficSpace

Definition:	An AuxiliaryTrafficSpace is a space within the transportation space not intended for traffic purposes.
Subclass Of:	AbstractUnoccupiedSpace
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
boundary	AuxiliaryTrafficArea [*]	

Attribute	Value type and multiplicity	Definition
class	AuxiliaryTrafficSpaceClassValue [0..1]	Indicates the specific type of the AuxiliaryTrafficSpace.
function	AuxiliaryTrafficSpaceFunctionValue [0..*]	Specifies the intended purposes of the AuxiliaryTrafficSpace.
granularity	GranularityValue	Defines whether auxiliary traffic spaces are represented by individual ways or by individual lanes, depending on the desired level of detail.
usage	AuxiliaryTrafficSpaceUsageValue [0..*]	Specifies the actual uses of the AuxiliaryTrafficSpace.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.7. Class AuxiliaryTrafficSpaceClassValue

AuxiliaryTrafficSpaceClassValue

Definition:	AuxiliaryTrafficSpaceClassValue is a code list used to further classify an AuxiliaryTrafficSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.8. Class AuxiliaryTrafficSpaceFunctionValue

AuxiliaryTrafficSpaceFunctionValue

Definition:	AuxiliaryTrafficSpaceFunctionValue is a code list that enumerates the different purposes of an AuxiliaryTrafficSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.9. Class AuxiliaryTrafficSpaceUsageValue

AuxiliaryTrafficSpaceUsageValue

Definition:	AuxiliaryTrafficSpaceUsageValue is a code list that enumerates the different uses of an AuxiliaryTrafficSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.10. Class ClearanceSpace

ClearanceSpace

Definition:	A ClearanceSpace represents the actual free space above a TrafficArea within which a mobile object can move without contacting an obstruction.
Subclass Of:	AbstractUnoccupiedSpace
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	ClearanceSpaceClassValue [0..*]	Indicates the specific type of the ClearanceSpace.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.11. Class ClearanceSpaceClassValue

ClearanceSpaceClassValue	
Definition:	ClearanceSpaceClassValue is a code list used to further classify a ClearanceSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.12. Class Hole

Hole	
Definition:	A Hole is an opening in the surface of a Road, Track or Square such as road damages, manholes or drains. Holes can span multiple transportation objects.
Subclass Of:	AbstractUnoccupiedSpace
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
boundary	AbstractThematicSurface [*]	

Attribute	Value type and multiplicity	Definition
class	HoleClassValue [0..1]	Indicates the specific type of the Hole.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.13. Class HoleClassValue

HoleClassValue	
Definition:	HoleClassValue is a code list used to further classify a Hole.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.14. Class HoleSurface

HoleSurface	
Definition:	A HoleSurface is a representation of the ground surface of a hole.
Subclass Of:	AbstractThematicSurface
StereoType:	«FeatureType»

9.1.27.15. Class Intersection

Intersection	
Definition:	An Intersection is a transportation space that is a shared segment of multiple Road, Track, Railway, or Waterway objects (e.g. a crossing of two roads or a level crossing of a road and a railway).
Subclass Of:	AbstractTransportationSpace
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	IntersectionClassValue [0..1]	Indicates the specific type of the Intersection.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.16. Class IntersectionClassValue

IntersectionClassValue	
Definition:	IntersectionClassValue is a code list used to further classify an Intersection.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.17. Class Marking

Marking

Definition:	A Marking is a visible pattern on a transportation area relevant to the structuring or restriction of traffic. Examples are road markings and markings related to railway or waterway traffic.
Subclass Of:	AbstractThematicSurface
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	MarkingClassValue [0..1]	Indicates the specific type of the Marking.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.18. Class MarkingClassValue

MarkingClassValue

Definition:	MarkingClassValue is a code list used to further classify a Marking.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.19. Class Railway

Railway

Definition:	A Railway is a transportation space used by wheeled vehicles on rails.
Subclass Of:	AbstractTransportationSpace
StereoType:	«TopLevelFeatureType»

Role name	Target class and multiplicity	Definition
section	Section [*]	
intersection	Intersection [*]	

Attribute	Value type and multiplicity	Definition
class	RailwayClassValue [0..1]	Indicates the specific type of the Railway.
function	RailwayFunctionValue [0..*]	Specifies the intended purposes of the Railway.
usage	RailwayUsageValue [0..*]	Specifies the actual uses of the Railway.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.20. Class RailwayClassValue

RailwayClassValue

Definition:	RailwayClassValue is a code list used to further classify a Railway.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.21. Class RailwayFunctionValue

RailwayFunctionValue

Definition:	RailwayFunctionValue is a code list that enumerates the different purposes of a Railway.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.22. Class RailwayUsageValue

RailwayUsageValue

Definition:	RailwayUsageValue is a code list that enumerates the different uses of a Railway.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.23. Class Road

Road

Definition:	A Road is a transportation space used by vehicles, bicycles and/or pedestrians.
Subclass Of:	AbstractTransportationSpace
StereoType:	«TopLevelFeatureType»

Role name	Target class and multiplicity	Definition
intersection	Intersection [*]	
section	Section [*]	

Attribute	Value type and multiplicity	Definition
class	RoadClassValue [0..1]	Indicates the specific type of the Road.
function	RoadFunctionValue [0..*]	Specifies the intended purposes of the Road.
usage	RoadUsageValue [0..*]	Specifies the actual uses of the Road.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.24. Class RoadClassValue

RoadClassValue

Definition:	RoadClassValue is a code list used to further classify a Road.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.25. Class RoadFunctionValue

RoadFunctionValue

Definition:	RoadFunctionValue is a code list that enumerates the different purposes of a Road.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.26. Class RoadUsageValue

RoadUsageValue	
Definition:	RoadUsageValue is a code list that enumerates the different uses of a Road.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.27. Class Section

Section	
Definition:	A Section is a transportation space that is a segment of a Road, Railway, Track, or Waterway.
Subclass Of:	AbstractTransportationSpace
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	SectionClassValue [0..1]	Indicates the specific type of the Section.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.28. Class SectionClassValue

SectionClassValue	
Definition:	SectionClassValue is a code list used to further classify a Section.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.29. Class Square

Square

Definition:	A Square is a transportation space for unrestricted movement for vehicles, bicycles and/or pedestrians. This includes plazas as well as large sealed surfaces such as parking lots.
Subclass Of:	AbstractTransportationSpace
StereoType:	«TopLevelFeatureType»

Attribute	Value type and multiplicity	Definition
class	SquareClassValue [0..1]	Indicates the specific type of the Square.
function	SquareFunctionValue [0..*]	Specifies the intended purposes of the Square.
usage	SquareUsageValue [0..*]	Specifies the actual uses of the Square.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.30. Class SquareClassValue

SquareClassValue

Definition:	SquareClassValue is a code list used to further classify a Square.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.31. Class SquareFunctionValue

SquareFunctionValue

Definition:	SquareFunctionValue is a code list that enumerates the different purposes of a Square.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.32. Class SquareUsageValue

SquareUsageValue

Definition:	SquareUsageValue is a code list that enumerates the different uses of a Square.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.33. Class SurfaceMaterialValue

SurfaceMaterialValue	
Definition:	SurfaceMaterialValue is a code list that enumerates the different surface materials.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.34. Class Track

Track	
Definition:	A Track is a small path mainly used by pedestrians. Tracks can be segmented into Sections and Intersections.
Subclass Of:	AbstractTransportationSpace
StereoType:	«TopLevelFeatureType»

Role name	Target class and multiplicity	Definition
section	Section [*]	
intersection	Intersection [*]	

Attribute	Value type and multiplicity	Definition
class	TrackClassValue [0..1]	Indicates the specific type of the Track.
function	TrackFunctionValue [0..*]	Specifies the intended purposes of the Track.
usage	TrackUsageValue [0..*]	Specifies the actual uses of the Track.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.35. Class TrackClassValue

TrackClassValue	
Definition:	TrackClassValue is a code list used to further classify a Track.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.36. Class TrackFunctionValue

TrackFunctionValue	
Definition:	TrackFunctionValue is a code list that enumerates the different purposes of a Track.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.37. Class TrackUsageValue

TrackUsageValue	
Definition:	TrackUsageValue is a code list that enumerates the different uses of a Track.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.38. Class TrafficArea

TrafficArea	
Definition:	A TrafficArea is the ground surface of a TrafficSpace. Traffic areas are the surfaces upon which traffic actually takes place.
Subclass Of:	AbstractThematicSurface
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	TrafficAreaClassValue [0..1]	Indicates the specific type of the TrafficArea.
function	TrafficAreaFunctionValue [0..*]	Specifies the intended purposes of the TrafficArea.
surfaceMaterial	SurfaceMaterialValue [0..1]	Specifies the type of pavement of the TrafficArea.
usage	TrafficAreaUsageValue [0..*]	Specifies the actual uses of the TrafficArea.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.39. Class TrafficAreaClassValue

TrafficAreaClassValue	
Definition:	TrafficAreaClassValue is a code list used to further classify a TrafficArea.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.40. Class TrafficAreaFunctionValue

TrafficAreaFunctionValue	
Definition:	TrafficAreaFunctionValue is a code list that enumerates the different purposes of a TrafficArea.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.41. Class TrafficAreaUsageValue

TrafficAreaUsageValue	
Definition:	TrafficAreaUsageValue is a code list that enumerates the different uses of a TrafficArea.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.42. Class TrafficSpace

TrafficSpace	
Definition:	A TrafficSpace is a space in which traffic takes place. Traffic includes the movement of entities such as trains, vehicles, pedestrians, ships, or other transportation types.
Subclass Of:	AbstractUnoccupiedSpace
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
clearanceSpace	ClearanceSpace [*]	
successor	TrafficSpace [*]	
boundary	TrafficArea [*]	
predecessor	TrafficSpace [*]	

Attribute	Value type and multiplicity	Definition
class	TrafficSpaceClassValue [0..1]	Indicates the specific type of the TrafficSpace.
function	TrafficSpaceFunctionValue [0..*]	Specifies the intended purposes of the TrafficSpace.
granularity	GranularityValue	Defines whether traffic spaces are represented by individual ways or by individual lanes, depending on the desired level of detail.
occupancy	Occupancy [0..*]	Provides information on the residency of persons, vehicles, or other moving features in the TrafficSpace.
trafficDirection	TrafficDirectionValue [0..1]	Indicates the direction of traffic flow relative to the corresponding linear geometry representation.
usage	TrafficSpaceUsageValue [0..*]	Specifies the actual uses of the TrafficSpace.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.43. Class TrafficSpaceClassValue

TrafficSpaceClassValue

Definition:	TrafficSpaceClassValue is a code list used to further classify a TrafficSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.44. Class TrafficSpaceFunctionValue

TrafficSpaceFunctionValue	
Definition:	TrafficSpaceFunctionValue is a code list that enumerates the different purposes of a TrafficSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.45. Class TrafficSpaceUsageValue

TrafficSpaceUsageValue	
Definition:	TrafficSpaceUsageValue is a code list that enumerates the different uses of a TrafficSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.46. Class TransportationSpaceClassValue

TransportationSpaceClassValue	
Definition:	TransportationSpaceClassValue is a code list used to further classify a TransportationSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.47. Class TransportationSpaceFunctionValue

TransportationSpaceFunctionValue

Definition:	TransportationSpaceFunctionValue is a code list that enumerates the different purposes of a TransportationSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.48. Class TransportationSpaceUsageValue

TransportationSpaceUsageValue	
Definition:	TransportationSpaceUsageValue is a code list that enumerates the different uses of a TransportationSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.49. Class Waterway

Waterway	
Definition:	A Waterway is a transportation space used for the movement of vessels upon or within a water body.
Subclass Of:	AbstractTransportationSpace
StereoType:	«TopLevelFeatureType»

Role name	Target class and multiplicity	Definition
section	Section [*]	
intersection	Intersection [*]	

Attribute	Value type and multiplicity	Definition
class	WaterwayClassValue [0..1]	Indicates the specific type of the Waterway.
function	WaterwayFunctionValue [0..*]	Specifies the intended purposes of the Waterway.
usage	WaterwayUsageValue [0..*]	Specifies the actual uses of the Waterway.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.50. Class WaterwayClassValue

WaterwayClassValue	
Definition:	WaterwayClassValue is a code list used to further classify a Waterway.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.51. Class WaterwayFunctionValue

WaterwayFunctionValue	
Definition:	WaterwayFunctionValue is a code list that enumerates the different purposes of a Waterway.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.52. Class WaterwayUsageValue

WaterwayUsageValue	
Definition:	WaterwayUsageValue is a code list that enumerates the different uses of a Waterway.
Subclass Of:	None
StereoType:	«CodeList»

9.1.27.53. Class GranularityValue

GranularityValue	
Definition:	GranularityValue enumerates the different levels of granularity in which transportation objects are represented.
Subclass Of:	None
StereoType:	

Attribute	Value type and multiplicity	Definition
lane		Indicates that the individual lanes of the transportation object are represented.
way		Indicates that the individual (carriage)ways of the transportation object are represented.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.27.54. Class TrafficDirectionValue

TrafficDirectionValue

Definition:	TrafficDirectionValue enumerates the allowed directions of travel of a mobile object.
Subclass Of:	None
StereoType:	

Attribute	Value type and multiplicity	Definition
forwards		Indicates that traffic flows in the direction of the corresponding linear geometry.
backwards		Indicates that traffic flows in the opposite direction of the corresponding linear geometry.
both		Indicates that traffic flows in both directions.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.28. Package

Package Tunnel

Description:	The Tunnel module supports representation of thematic and spatial aspects of tunnels, tunnel parts, tunnel installations, and interior tunnel structures.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.29. Classes

9.1.29.1. Class AbstractTunnel

AbstractTunnel	
Definition:	AbstractTunnel is an abstract superclass representing the common attributes and associations of the classes Tunnel and TunnelPart.
Subclass Of:	AbstractConstruction
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
tunnelFurniture	TunnelFurniture [*]	
tunnelInstallation	TunnelInstallation [*]	
tunnelConstructiveElement	TunnelConstructiveElement [*]	
hollowSpace	HollowSpace [*]	

Attribute	Value type and multiplicity	Definition
class	TunnelClassValue [0..1]	Indicates the specific type of the Tunnel or TunnelPart.
function	TunnelFunctionValue [0..*]	Specifies the intended purposes of the Tunnel or TunnelPart.
usage	TunnelUsageValue [0..*]	Specifies the actual uses of the Tunnel or TunnelPart.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.29.2. Class HollowSpace

HollowSpace	
Definition:	A HollowSpace is a space within a Tunnel or TunnelPart intended for certain functions (e.g. transport or passage ways, service rooms, emergency shelters). A HollowSpace is bounded physically and/or virtually (e.g. by ClosureSurfaces or GenericSurfaces).
Subclass Of:	AbstractUnoccupiedSpace
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
boundary	AbstractThematicSurface [*]	
tunnelFurniture	TunnelFurniture [*]	
tunnelInstallation	TunnelInstallation [*]	

Attribute	Value type and multiplicity	Definition
class	HollowSpaceClassValue [0..1]	Indicates the specific type of the HollowSpace.
function	HollowSpaceFunctionValue [0..*]	Specifies the intended purposes of the HollowSpace.
usage	HollowSpaceUsageValue [0..*]	Specifies the actual uses of the HollowSpace.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.29.3. Class HollowSpaceClassValue

HollowSpaceClassValue	
Definition:	HollowSpaceClassValue is a code list used to further classify a HollowSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.4. Class HollowSpaceFunctionValue

HollowSpaceFunctionValue	
Definition:	HollowSpaceFunctionValue is a code list that enumerates the different purposes of a HollowSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.5. Class HollowSpaceUsageValue

HollowSpaceUsageValue	
Definition:	HollowSpaceUsageValue is a code list that enumerates the different uses of a HollowSpace.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.6. Class Tunnel

Tunnel	
Definition:	A Tunnel represents a horizontal or sloping enclosed passage way of a certain length, mainly underground or underwater. [cf. ISO 6707-1]
Subclass Of:	AbstractTunnel
StereoType:	«TopLevelFeatureType»

Role name	Target class and multiplicity	Definition
tunnelPart	TunnelPart [*]	

9.1.29.7. Class TunnelClassValue

TunnelClassValue	
Definition:	TunnelClassValue is a code list used to further classify a Tunnel.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.8. Class TunnelConstructiveElement

TunnelConstructiveElement	
Definition:	A TunnelConstructiveElement is an element of a Tunnel which is essential from a structural point of view. Examples are walls, slabs, beams.
Subclass Of:	AbstractConstructiveElement
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	TunnelConstructiveElementClassValue [0..1]	Indicates the specific type of the TunnelConstructiveElement.
function	TunnelConstructiveElementFunctionValue [0..*]	Specifies the intended purposes of the TunnelConstructiveElement.
usage	TunnelConstructiveElementUsageValue [0..*]	Specifies the actual uses of the TunnelConstructiveElement.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.29.9. Class TunnelConstructiveElementClassValue

TunnelConstructiveElementClassValue	
Definition:	TunnelConstructiveElementClassValue is a code list used to further classify a TunnelConstructiveElement.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.10. Class TunnelConstructiveElementFunctionValue

TunnelConstructiveElementFunctionValue	
Definition:	TunnelConstructiveElementFunctionValue is a code list that enumerates the different purposes of a TunnelConstructiveElement.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.11. Class TunnelConstructiveElementUsageValue

TunnelConstructiveElementUsageValue

Definition:	TunnelConstructiveElementUsageValue is a code list that enumerates the different uses of a TunnelConstructiveElement.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.12. Class TunnelFunctionValue

TunnelFunctionValue	
Definition:	TunnelFunctionValue is a code list that enumerates the different purposes of a Tunnel.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.13. Class TunnelFurniture

TunnelFurniture	
Definition:	A TunnelFurniture is an equipment for occupant use, usually not fixed to the tunnel. [cf. ISO 6707-1]
Subclass Of:	AbstractFurniture
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	TunnelFurnitureClassValue [0..1]	Indicates the specific type of the TunnelFurniture.
function	TunnelFurnitureFunctionValue [0..*]	Specifies the intended purposes of the TunnelFurniture.
usage	TunnelFurnitureUsageValue [0..*]	Specifies the actual uses of the TunnelFurniture.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.29.14. Class TunnelFurnitureClassValue

TunnelFurnitureClassValue

Definition:	TunnelFurnitureClassValue is a code list used to further classify a TunnelFurniture.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.15. Class TunnelFurnitureFunctionValue

TunnelFurnitureFunctionValue	
Definition:	TunnelFurnitureFunctionValue is a code list that enumerates the different purposes of a TunnelFurniture.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.16. Class TunnelFurnitureUsageValue

TunnelFurnitureUsageValue	
Definition:	TunnelFurnitureUsageValue is a code list that enumerates the different uses of a TunnelFurniture.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.17. Class TunnelInstallation

TunnelInstallation	
Definition:	A TunnelInstallation is a permanent part of a Tunnel (inside and/or outside) which does not have the significance of a TunnelPart. In contrast to TunnelConstructiveElement, a TunnelInstallation is not essential from a structural point of view. Examples are stairs, antennas or railings.
Subclass Of:	AbstractInstallation
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
class	TunnelInstallationClassValue [0..1]	Indicates the specific type of the TunnelInstallation.
function	TunnelInstallationFunctionValue [0..*]	Specifies the intended purposes of the TunnelInstallation.
usage	TunnelInstallationUsageValue [0..*]	Specifies the actual uses of the TunnelInstallation.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.29.18. Class TunnelInstallationClassValue

TunnelInstallationClassValue	
Definition:	TunnelInstallationClassValue is a code list used to further classify a TunnelInstallation.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.19. Class TunnelInstallationFunctionValue

TunnelInstallationFunctionValue	
Definition:	TunnelInstallationFunctionValue is a code list that enumerates the different purposes of a TunnelInstallation.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.20. Class TunnelInstallationUsageValue

TunnelInstallationUsageValue	
Definition:	TunnelInstallationUsageValue is a code list that enumerates the different uses of a TunnelInstallation.
Subclass Of:	None
StereoType:	«CodeList»

9.1.29.21. Class TunnelPart

TunnelPart	
Definition:	A TunnelPart is a physical or functional subdivision of a Tunnel. It would be considered a Tunnel, if it were not part of a collection of other TunnelParts.
Subclass Of:	AbstractTunnel
StereoType:	«FeatureType»

9.1.29.22. Class TunnelUsageValue

TunnelUsageValue	
Definition:	TunnelUsageValue is a code list that enumerates the different uses of a Tunnel.
Subclass Of:	None
StereoType:	«CodeList»

9.1.30. Package

Package Vegetation	
Description:	The Vegetation module supports representation of vegetation objects with vegetation-specific thematic classes. CityGML's vegetation model distinguishes between solitary vegetation objects like trees, and vegetation areas which represent biotopes like forests or other plant communities.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.31. Classes

9.1.31.1. Class AbstractVegetationObject

AbstractVegetationObject	
Definition:	AbstractVegetationObject is the abstract superclass for all kinds of vegetation objects.
Subclass Of:	AbstractOccupiedSpace
StereoType:	«FeatureType»

9.1.31.2. Class PlantCover

PlantCover	
Definition:	A PlantCover represents a space covered by vegetation.
Subclass Of:	AbstractVegetationObject
StereoType:	«TopLevelFeatureType»

Attribute	Value type and multiplicity	Definition
averageHeight	Length [0..1]	Specifies the average height of the PlantCover.
class	PlantCoverClassValue [0..1]	Indicates the specific type of the PlantCover.
function	PlantCoverFunctionValue [0..*]	Specifies the intended purposes of the PlantCover.
maxHeight	Length [0..1]	Specifies the maximum height of the PlantCover.
minHeight	Length [0..1]	Specifies the minimum height of the PlantCover.
usage	PlantCoverUsageValue [0..*]	Specifies the actual uses of the PlantCover.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.31.3. Class PlantCoverClassValue

PlantCoverClassValue	
Definition:	PlantCoverClassValue is a code list used to further classify a PlantCover.
Subclass Of:	None
StereoType:	«CodeList»

9.1.31.4. Class PlantCoverFunctionValue

PlantCoverFunctionValue	
Definition:	PlantCoverFunctionValue is a code list that enumerates the different purposes of a PlantCover.
Subclass Of:	None
StereoType:	«CodeList»

9.1.31.5. Class PlantCoverUsageValue

PlantCoverUsageValue	
Definition:	PlantCoverUsageValue is a code list that enumerates the different uses of a PlantCover.
Subclass Of:	None
StereoType:	«CodeList»

9.1.31.6. Class SolitaryVegetationObject

SolitaryVegetationObject	
Definition:	A SolitaryVegetationObject represents individual vegetation objects, e.g. trees or bushes.
Subclass Of:	AbstractVegetationObject
StereoType:	«TopLevelFeatureType»

Attribute	Value type and multiplicity	Definition
class	SolitaryVegetationObjectClassValue [0..1]	Indicates the specific type of the SolitaryVegetationObject.
crownDiameter	Length [0..1]	Specifies the diameter of the SolitaryCityObject's crown.
function	SolitaryVegetationObjectFunctionValue [0..*]	Specifies the intended purposes of the SolitaryVegetationObject.
height	Length [0..1]	Distance between the highest point of the vegetation object and the lowest point of the terrain at the bottom of the object.
maxRootBallDepth	Length [0..1]	Specifies the vertical distance between the lowest point of the SolitaryVegetationObject's root ball and the terrain surface.
rootBallDiameter	Length [0..1]	Specifies the diameter of the SolitaryCityObject's root ball.
species	SpeciesValue [0..1]	Indicates the botanical name of the SolitaryVegetationObject.
trunkDiameter	Length [0..1]	Specifies the diameter of the SolitaryCityObject's trunk.
usage	SolitaryVegetationObjectUsageValue [0..*]	Specifies the actual uses of the SolitaryVegetationObject.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.31.7. Class SolitaryVegetationObjectClassValue

SolitaryVegetationObjectClassValue	
Definition:	SolitaryVegetationObjectClassValue is a code list used to further classify a SolitaryVegetationObject.
Subclass Of:	None
StereoType:	«CodeList»

9.1.31.8. Class SolitaryVegetationObjectFunctionValue

SolitaryVegetationObjectFunctionValue

Definition:	SolitaryVegetationObjectFunctionValue is a code list that enumerates the different purposes of a SolitaryVegetationObject.
Subclass Of:	None
StereoType:	«CodeList»

9.1.31.9. Class SolitaryVegetationObjectUsageValue

SolitaryVegetationObjectUsageValue	
Definition:	SolitaryVegetationObjectUsageValue is a code list that enumerates the different uses of a SolitaryVegetationObject.
Subclass Of:	None
StereoType:	«CodeList»

9.1.31.10. Class SpeciesValue

SpeciesValue	
Definition:	A SpeciesValue is a code list that enumerates the species of a SolitaryVegetationObject.
Subclass Of:	None
StereoType:	«CodeList»

9.1.32. Package

Package Versioning	
Description:	The Versioning module supports representation of multiple versions of CityGML features within a single CityGML model. In addition, also the version transitions and transactions that lead to the different versions can be represented.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.33. Classes

9.1.33.1. Class Version

Version		
Definition:	Version represents a defined state of a city model consisting of the dedicated versions of all city object instances that belong to the respective city model version. Versions can have names, a description and can be labeled with an arbitrary number of user defined tags.	
Subclass Of:	AbstractVersion	
StereoType:	«FeatureType»	

Role name	Target class and multiplicity	Definition
versionMember	AbstractFeatureWithLifespan [*]	

Attribute	Value type and multiplicity	Definition
tag	CharacterString [0..*]	Allows for adding keywords to the city model version.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.33.2. Class VersionTransition

VersionTransition	
Definition:	VersionTransition describes the change of the state of a city model from one version to another. Version transitions can have names, a description and can be further qualified by a type and a reason.
Subclass Of:	AbstractVersionTransition
StereoType:	«FeatureType»

Role name	Target class and multiplicity	Definition
transaction	Transaction [*]	
from	Version [0..1]	
to	Version [0..1]	

Attribute	Value type and multiplicity	Definition
clonePredecessor	Boolean	Indicates whether the set of city object instances belonging to the successor version of the city model is either explicitly enumerated within the successor version object (attribute clonePredecessor=false), or has to be derived from the modifications of the city model provided as a list of transactions on the city object versions contained in the predecessor version (attribute clonePredecessor=true).
reason	CharacterString [0..1]	Specifies why the VersionTransition has been carried out.
type	TransitionTypeValue [0..1]	Indicates the specific type of the VersionTransition.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.33.3. Class Transaction

Transaction

Definition:	Transaction represents a modification of the city model by the creation, termination, or replacement of a specific city object. While the creation of a city object also marks its first object version, the termination marks the end of existence of a real world object and, hence, also terminates the final version of a city object. The replacement of a city object means that a specific version of it is replaced by a new version.
Subclass Of:	<←section,>>
StereoType:	«DataType»

Role name	Target class and multiplicity	Definition
newFeature	AbstractFeatureWithLifespan [0..1]	
oldFeature	AbstractFeatureWithLifespan [0..1]	

Attribute	Value type and multiplicity	Definition
type	TransactionTypeValue	Indicates the specific type of the Transaction.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.33.4. Class TransactionTypeValue

TransactionTypeValue	
Definition:	TransactionTypeValue enumerates the three possible types of transactions: insert, delete, or replace.
Subclass Of:	None
StereoType:	

Attribute	Value type and multiplicity	Definition
insert		Indicates that the feature referenced from the Transaction via the "newFeature" association has been newly created; the association "oldFeature" is empty in this case.
delete		Indicates that the feature referenced from the Transaction via the "oldFeature" association ceases to exist; the association "newFeature" is empty in this case.
replace		Indicates that the feature referenced from the Transaction via the "oldFeature" association has been replaced by the feature referenced via the "newFeature" association.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.33.5. Class TransitionTypeValue

TransitionTypeValue	
Definition:	TransitionTypeValue enumerates the different kinds of version transitions. “planned” and “fork” should be used in cases when from one city model version multiple successor versions are being created. “realized” and “merge” should be used when different city model versions are converging into a common successor version.
Subclass Of:	None
StereoType:	

Attribute	Value type and multiplicity	Definition
planned		Indicates that the successor version of the city model represents a planning state for a possible future of the city.
realized		Indicates that the predecessor version is the chosen one from a number of possible planning versions.
historicalSuccession		Indicates that the successor version reflects updates on the city model over time (historical timeline). It shall only be used for at most one version transition outgoing from a city model version.
fork		Indicates other reasons to create alternative city model versions, for example, when different parties are updating parts of the city model or to reflect the results of different simulation runs.
merge		Indicates other reasons to converge multiple versions back into a common city model version.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.34. Package

Package WaterBody

Description:	The WaterBody module supports representation of the thematic aspects and 3D geometry of rivers, canals, lakes, and basins. It does, however, not inherit any hydrological or other dynamic aspects of fluid flow.
Parent Package:	CityGML
Stereotype:	«ApplicationSchema»

9.1.35. Classes

9.1.35.1. Class AbstractWaterBoundarySurface

AbstractWaterBoundarySurface

Definition:	AbstractWaterBoundarySurface is the abstract superclass for all kinds of thematic surfaces bounding a water body.
Subclass Of:	AbstractThematicSurface
StereoType:	«FeatureType»

9.1.35.2. Class WaterBody

WaterBody	
Definition:	A WaterBody represents significant and permanent or semi-permanent accumulations of surface water, usually covering a part of the Earth.
Subclass Of:	AbstractOccupiedSpace
StereoType:	«TopLevelFeatureType»

Role name	Target class and multiplicity	Definition
boundary	AbstractWaterBoundarySurface [*]	

Attribute	Value type and multiplicity	Definition
class	WaterBodyClassValue [0..1]	Indicates the specific type of the WaterBody.
function	WaterBodyFunctionValue [0..*]	Specifies the intended purposes of the WaterBody.
usage	WaterBodyUsageValue [0..*]	Specifies the actual uses of the WaterBody.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

9.1.35.3. Class WaterBodyClassValue

WaterBodyClassValue	
Definition:	WaterBodyClassValue is a code list used to further classify a WaterBody.
Subclass Of:	None
StereoType:	«CodeList»

9.1.35.4. Class WaterBodyFunctionValue

WaterBodyFunctionValue

Definition:	WaterBodyFunctionValue is a code list that enumerates the different purposes of a WaterBody.
Subclass Of:	None
StereoType:	«CodeList»

9.1.35.5. Class WaterBodyUsageValue

WaterBodyUsageValue	
Definition:	WaterBodyUsageValue is a code list that enumerates the different uses of a WaterBody.
Subclass Of:	None
StereoType:	«CodeList»

9.1.35.6. Class WaterGroundSurface

WaterGroundSurface	
Definition:	A WaterGroundSurface represents the exterior boundary surface of the submerged bottom of a water body.
Subclass Of:	AbstractWaterBoundarySurface
StereoType:	«FeatureType»

9.1.35.7. Class WaterLevelValue

WaterLevelValue	
Definition:	WaterLevelValue is a code list that enumerates the different levels of a water surface.
Subclass Of:	None
StereoType:	«CodeList»

9.1.35.8. Class WaterSurface

WaterSurface

Definition:	A WaterSurface represents the upper exterior interface between a water body and the atmosphere.
Subclass Of:	AbstractWaterBoundarySurface
StereoType:	«FeatureType»

Attribute	Value type and multiplicity	Definition
waterLevel	WaterLevelValue [0..1]	Specifies the level of the WaterSurface.

Note: Unless otherwise specified, all attributes have the stereotype «Property»

Chapter 10. Media Types for any data encoding(s)

A section describing the MIME-types to be used is mandatory for any standard involving data encodings. If no suitable MIME type exists in http://www.iana.org/assignments/media-types/index.html then this section may be used to define a new MIME type for registration with IANA.

Annex A: Conformance Class Abstract Test Suite (Normative)

Annex B: Revision History

Date	Release	Editor	Primary clauses modified	Description
2016-04-28	0.1	G. Editor	all	initial version

Chapter 11. Change Log for CityGML 3.0

CityGML 3.0 is the baseline version. Changes appearing in versions following 3.0 will be listed in this section.

Feature Class / Data Type	Property	New	Changed	Deleted	Description of Change

Annex C: Changelog for CityGML 3.0

The following table lists all feature types, properties, and data types which have been added or changed for CityGML 3.0.

Feature Class / Data Type	Property	New	Changed	Deleted	Description of Change

Annex D: Bibliography

NOTE

Example Bibliography (Delete this note).

The TC has approved Springer LNCS as the official document citation type.

Springer LNCS is widely used in technical and computer science journals and other publications

- For citations in the text please use square brackets and consecutive numbers: [1], [2], [3]

– Actual References:

[n] Journal: Author Surname, A.: Title. Publication Title. Volume number, Issue number, Pages Used (Year Published)

[n] Web: Author Surname, A.: Title, http://Website-Url

[1] OGC: OGC Testbed 12 Annex B: Architecture. (2015).