

# **Open Geospatial Consortium**

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: <2020-02-24>

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: 0.5

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**

### **Copyright notice**

Copyright © 2020 Open Geospatial Consortium

To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>

### **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

1. Introduction .....	10
1.1. Motivation .....	10
1.2. Historical background .....	10
1.3. Additions in CityGML 2.0 .....	12
2. Scope .....	15
3. Conformance .....	17
4. References .....	18
5. Terms and Definitions .....	20
5.1. Abbreviated Terms .....	22
6. Conventions .....	24
6.1. Identifiers .....	24
6.2. UML Notation .....	24
6.3. XML namespaces and namespace prefixes .....	26
7. Overview of CityGML .....	29
8. General characteristics of CityGML .....	31
8.1. Modularisation .....	31
8.2. Multi-scale modelling (5 levels of detail, LOD) .....	31
8.3. Coherent semantical-geometrical modelling .....	33
8.4. Closure surfaces .....	34
8.5. Terrain Intersection Curve (TIC) .....	35
8.6. Code lists for enumerative attributes .....	37
8.7. External references .....	38
8.8. City object groups .....	39
8.9. Appearances .....	39
8.10. Prototypic objects / scene graph concepts .....	40
8.11. Generic city objects and attributes .....	41
8.12. Application Domain Extensions (ADE) .....	41
9. Modularization .....	43
9.1. CityGML core and extension modules .....	44
9.2. CityGML profiles .....	50
10. Spatial Model .....	52
10.1. Geometric-topological model .....	52
10.2. Spatial Reference System .....	55
10.3. Implicit geometries, prototypic objects, scene graph concepts .....	55
11. CityGML Conceptual Model .....	56
11.1. Core .....	56
11.2. Appearance .....	94
11.3. Bridge .....	109

11.4. Building .....	125
11.5. City Furniture .....	148
11.6. Module City Furniture new 1 .....	152
11.7. Module City Furniture new 2 .....	156
11.8. City Object Group .....	159
11.9. Construction .....	163
11.10. Dynamizer .....	191
11.11. Generics .....	210
11.12. Land Use .....	227
11.13. Point Cloud .....	231
11.14. Digital Terrain Model .....	233
11.15. Transportation .....	240
11.16. Tunnel .....	275
11.17. Vegetation .....	290
11.18. Versioning .....	298
11.19. Water Body .....	305
12. Media Types for any data encoding(s) .....	312
Annex A: Conformance Class Abstract Test Suite (Normative) .....	313
A.1. Conformance Class Appearance .....	313
A.2. Conformance Class Bridge .....	322
A.3. Conformance Class Building .....	331
A.4. Conformance Class CityFurniture .....	345
A.5. Conformance Class CityObjectGroup .....	346
A.6. Conformance Class Core .....	349
A.7. Conformance Class Dynamizer .....	374
A.8. Conformance Class Generics .....	384
A.9. Conformance Class LandUse .....	395
A.10. Conformance Class PointCloud .....	397
A.11. Conformance Class Relief .....	398
A.12. Conformance Class Transportation .....	402
A.13. Conformance Class Tunnel .....	424
A.14. Conformance Class Vegetation .....	433
A.15. Conformance Class Versioning .....	437
A.16. Conformance Class WaterBody .....	441
Annex B: Revision History .....	446
13. Change Log for CityGML 3.0 .....	447
Annex C: Changelog for CityGML 3.0 .....	448
Annex D: Bibliography .....	449

## **i. Abstract**

CityGML is an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is an application schema for the Geography Markup Language version 3.2.1 (GML3), the extendible international standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211.

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

## **ii. Keywords**

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

ogcdoc, OGC document, <tags separated by commas>

## **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.

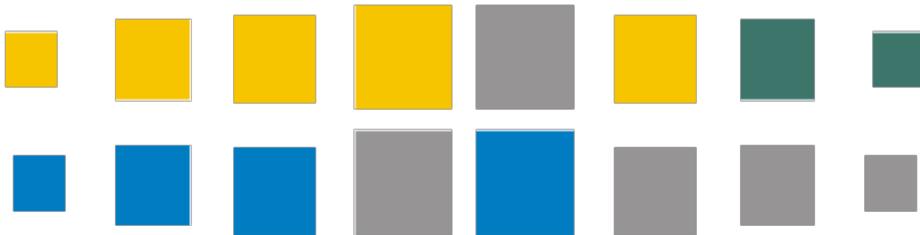
Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

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

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

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

## **iv. Submitting organizations**



# CityGML

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[]"

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

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

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



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

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



The preparation of the English document version and the European discussion has been supported by the European Spatial Data Research Organization (EuroSDR; formerly known as OEEPE) in an EuroSDR Commission III project. For further information see <http://www.eurosdr.net>

This Document was submitted to the Open Geospatial Consortium (OGC) by the members of the CityGML 3.0 Standards Working Group of the OGC. Amongst others, this comprises the following organizations:

- Autodesk, Inc. (primary submitter)
- Bentley Systems, Inc. (primary submitter)
- Technical University Berlin (submitter of technology)
- Ordnance Survey, UK
- University of Bonn, Germany
- Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam
- Institute for Applied Computer Science, Karlsruhe Institute of Technology

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

## v. Submitters

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

*Table 1. Submission Contact Points*

Name	Institution	Email
Prof. Dr. Thomas H. Kolbe, Claus Nagel, Alexandra Lorenz	Institute for Geodesy and Geoinformation Science, Technical University Berlin	<a href="mailto:thomas.kolbe@tu-berlin.de">thomas.kolbe@tu-berlin.de</a> <a href="mailto:claus.nagel@tu-berlin.de">claus.nagel@tu-berlin.de</a> <a href="mailto:alexandra.lorenz@tu-berlin.de">alexandra.lorenz@tu-berlin.de</a>
Dr. Gerhard Gröger, Prof. Dr. Lutz Plümer, Angela Czerwinski	Institute for Geodesy and Geoinformation, University of Bonn	<a href="mailto:Groeger@ikg.uni-bonn.de">Groeger@ikg.uni-bonn.de</a> <a href="mailto:Pluemer@ikg.uni-bonn.de">Pluemer@ikg.uni-bonn.de</a> <a href="mailto:Czerwinski@ikg.uni-bonn.de">Czerwinski@ikg.uni-bonn.de</a>
Haik Lorenz	Autodesk, Inc.	<a href="mailto:haik.lorenz@autodesk.com">haik.lorenz@autodesk.com</a>
Alain Lapierre, Stefan Apfel, Paul Scarponeini	Bentley Systems, Inc.	<a href="mailto:alain.lapierre@bentley.com">alain.lapierre@bentley.com</a> <a href="mailto:stefan.apfel@bentley.com">stefan.apfel@bentley.com</a> <a href="mailto:paul.scarponeini@bentley.com">paul.scarponeini@bentley.com</a>
Carsten Rönsdorf	Ordnance Survey, Great Britain	<a href="mailto:carsten.roensdorf@ordnancesurvey.co.uk">carsten.roensdorf@ordnancesurvey.co.uk</a>

Name	Institution	Email
Prof. Dr. Jürgen Döllner	Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam	<a href="mailto:juergen.doellner@hpi.uni-potsdam.de">juergen.doellner@hpi.uni-potsdam.de</a>
Dr. Joachim Benner, Karl-Heinz Häfele	Institute for Applied Computer Science, Karlsruhe Institute of Technology	<a href="mailto:joachim.benner@kit.edu">joachim.benner@kit.edu</a> <a href="mailto:karl-heinz.haefele@kit.edu">karl-heinz.haefele@kit.edu</a>

## vi. Participants in development

Table 2. Participants in Development

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

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

## vii. Acknowledgements

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

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

**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 pedestrian navigation, facility management and environmental simulations. Furthermore, in the implementation of the European Environmental Noise Directive (END, 2002/49/EC) 3D geoinformation and 3D city models play an important role.

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

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

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

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

## 1.2. Historical background

CityGML has been developed since 2002 by the members of the Special Interest Group 3D (SIG 3D). Since 2010, this group is part of the initiative Spatial Data Infrastructure Germany (GDI-DE). Before

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

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

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

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

## 1.3. Additions in CityGML 2.0

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

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

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

### New thematic modules for the representation of bridges and tunnels

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

Bridges and tunnels can be represented in LOD 1 – 4 and the underlying data models have a coherent structure with the Building model. For example, bridges and tunnels can be decomposed into parts, thematic boundary surfaces with openings are available to semantically classify parts of the shell, and installations as well as interi-or built structures can be represented. This coherent model structure facilitates the similar understanding of semantic entities and helps to reduce software implementation efforts. Both the Bridge and the Tunnel model introduce further concepts and model elements which are specific to bridges and tunnels respectively.

### Additions to existing thematic modules

- *CityGML Core module* (cf. chapter 10.1) Two new optional attributes have been added to the abstract base class *core:\_CityObject* within the *CityGML Core* module: *relativeToTerrain* and *relativeToWater*. These attributes denote the feature's location with respect to the terrain and water surface in a qualitative way, and thus facilitate simple and efficient queries (e.g., for the number of subsurface buildings) without the need for an additional digital terrain model or a model of the water body.
- *Building module* (cf. chapter 10.3)
  - *LOD0 representation* : Buildings can now be represented in LOD0 by footprint and/or roof

edge polygons. This allows the easy integration of existing 2D data and of roof reconstructions from aerial and satellite imagery into a 3D city model. The representations are restricted to horizontal, 3-dimensional surfaces.

- *Additional thematic boundary surfaces* : In order to semantically classify parts of the outer building shell which are neither horizontal wall surfaces nor parts of the roof, two additional boundary surfaces are introduced: *OuterFloorSurface* and *OuterCeilingSurface*.
- *Additional relations to thematic boundary surfaces* : In addition to *\_AbstractBuilding* and *Room*, the surface geometries of *BuildingInstallation* and *IntBuildingInstallation* features can now be semantically classified using thematic boundary surfaces. For example, this facilitates the semantic differentiation between roof and wall surfaces of dormers which are modeled as *BuildingInstallation*.
- *Additional use of implicit geometries* : Implicit geometries (cf. chapter 8.3) are now available for the representation of *\_Opening*, *BuildingInstallation*, and *IntBuildingInstallation* in addition to *BuildingFurniture*. A prototypical geometry for these city objects can thus be stored once and instantiated at different locations in the 3D city model.
- *Generics module* (cf. chapter 10.12) Two generic attributes have been added to the *Generics* module: *MeasureAttribute* and *GenericAttributeSet*. A *MeasureAttribute* facilitates the representation of measured values together with a reference to the employed unit. A *GenericAttributeSet* is a named collection of arbitrary generic attributes. It provides an optional *codeSpace* attribute to denote the authority organization who defined the attribute set.
- *LandUse module* (cf. chapter 10.10) The scope of the feature type *LandUse* has been broadened to comprise both areas of the earth's surface dedicated to a specific land use and areas of the earth's surface having a specific land cover with or without vegetation.
- *Attributes class, function, and usage (all modules)* In order to harmonize the use of the attributes *class*, *function*, and *usage*, this attribute triplet has been complemented for all feature classes that at least provided one of the attributes in CityGML 1.0.

## Additions to the CityGML code list mechanism

In CityGML, code lists providing the allowed values for enumerative attributes such as *class*, *function*, and *usage* can be specified outside the CityGML schema by any organization or information community according to their specific information needs. This mechanism is, however, not fully reflected in the CityGML 1.0 encoding schema, because in a CityGML 1.0 instance document a corresponding attribute cannot point to the dictionary with the used code list values. This has been corrected for CityGML 2.0: All attributes taking values from code lists are now of type *gml:CodeType* following the GML 3.1.1 mechanism for the encoding of code list values (cf. chapter 10.14 for more information). The *gml:CodeType* adds an optional *codeSpace* value to enumerative attributes which allows for providing a persistent URI pointing to the corresponding dictionary.

## Changelog for CityGML 2.0

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

## Further edits to the specification document

- *Accuracy requirements for Levels of Detail (LOD)* (cf. chapter 6.2) The accuracy requirements for the different CityGML LODs proposed in chapter 6.2 are non-normative. The wording of chapter

6.2 in CityGML 1.0 is however inconsistent with regard to this fact and thus has been clarified for CityGML 2.0.

- *Rework of the CityGML example datasets (cf. Annex G)* The CityGML examples provided in Annex G have been reworked and extended. They now show a consistent building model in all five LODs and demonstrate, for example, the semantic and geometric refinement of the building throughout the different LODs as well as the usage of XLinks to share geometry elements between features. The datasets are shipped with the CityGML XML Schema package, and are available at <http://schemas.opengis.net/citygml/examples/2.0/>.
- *New example for the usage of Application Domain Extensions (cf. Annex I)* A second example for the usage of Application Domain Extensions in the field of Ubiquitous Network Robots Services has been added in Annex I.

**NOTE** Explanatory text edit 01 for 01.04.2020 release ends here

**NOTE** Explanatory text edit 02 for 01.04.2020 release starts here

# Chapter 2. Scope

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

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

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

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

Features of CityGML:

- Geospatial information model (ontology) for urban landscapes based on the ISO 191xx family
- GML3 representation of 3D geometries, based on the ISO 19107 model
- Representation of object surface characteristics (e.g. textures, materials)
- Taxonomies and aggregations
  - Digital Terrain Models as a combination of (including nested) triangulated irregular networks (TINs), regular rasters, break and skeleton lines, mass points
  - Sites (currently buildings, bridges, and tunnels)
  - Vegetation (areas, volumes, and solitary objects with vegetation classification)
  - Water bodies (volumes, surfaces)
  - Transportation facilities (both graph structures and 3D surface data)
  - Land use (representation of areas of the earth's surface dedicated to a specific land use)
  - City furniture
  - Generic city objects and attributes
  - User-definable (recursive) grouping
- Multiscale model with 5 well-defined consecutive Levels of Detail (LOD):
  - LOD0 – regional, landscape
  - LOD1 – city, region

- LOD2 – city districts, projects
- LOD3 – architectural models 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 application specific extensions, for example for noise pollution simulation, or to augment CityGML by properties of the new National Building Information Model Standard (NBIMS) in the U.S.

**NOTE** Explanatory text edit 02 for 01.04.2020 release ends here

# Chapter 3. Conformance

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 implementing technologies, the specific techniques to be used for conformance testing cannot be specified. It is the responsibility 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 discrepancy between the UML model and this document, the UML model is authoritative.

**NOTE** Explanatory text edit 03 for 01.04.2020 release starts here

# 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.

- 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
- 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
- IETF: RFC 2045 & 2046, Multipurpose Internet Mail Extensions (MIME). (November 1996)
- IETF: RFC 3986, Uniform Resource Identifier (URI): Generic Syntax. (January 2005)
- OASIS: extensible Address Language (xAL v2.0)
- Khronos Group Inc.: COLLADA – Digital Asset Schema Release 1.5.0

**The following references were listed in the CityGML 2.0 specification. For CityGML 3.0, these references should either be moved to the GML Encoding specification and be updated there or be removed completely:**

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

- ISO / TC211: ISO/TS 19139:2007, Geographic Information – Metadata – XML schema implementation
- OGC: OpenGIS® Geography Markup Language Implementation Specification, Version 3.1.1, OGC document 03-105r1
- OGC: OpenGIS® GML 3.1.1 Simple Dictionary Profile, Version 1.0.0, OGC document 05-099r2
- W3C: W3C XLink, XML Linking Language (XLink) Version 1.0. W3C Recommendation (27 June 2001)
- W3C: W3C XMLName, Namespaces in XML. W3C Recommendation (14 January 1999)
- W3C: W3C XMLSchema-1, XML Schema Part 1: Structures. W3C Recommendation (2 May 2001)
- W3C: W3C XMLSchema-2, XML Schema Part 2: Datatypes. W3C Recommendation (2 May 2001)
- W3C: W3C XPointer, XML Pointer Language (XPointer) Version 1.0. W3C Working Draft (16 August 2002)
- W3C: W3C XML Base, XML Base, W3C Recommendation (27 June 2001)
- W3C: W3C XML, Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation (6 October 2000)
- Jelliffe, R: The Schematron Assertion Language 1.5. (2002-10-01)

**NOTE** Explanatory text edit 03 for 01.04.2020 release ends here

**NOTE** Explanatory text edit 04 for 01.04.2020 release starts here

# 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.

**I looked at several standards and copied some relevant definitions. The list needs to be reviewed once all sections have been updated. It needs to be checked, if all important terms are included, if all defined terms actually appear in the text, and if all defined terms are necessary. Also, the formatting should be improved.**

## 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

**NOTE** Explanatory text edit 04 for 01.04.2020 release ends here

**NOTE** Explanatory text edit 05 for 01.04.2020 release starts here

# Chapter 6. Conventions

## 6.1. Identifiers

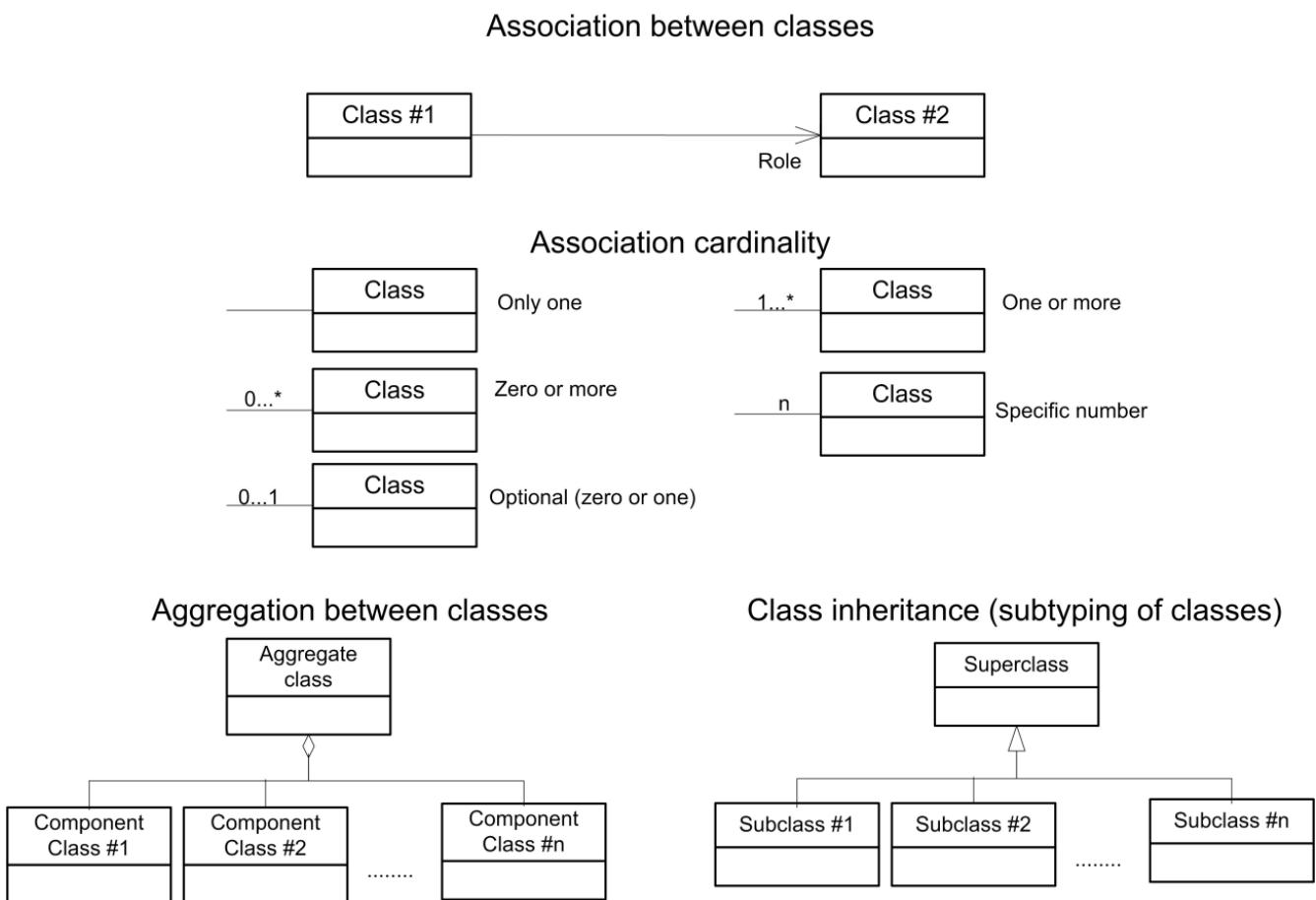
The normative provisions in this document are denoted by the URI

<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).*

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

The following stereotypes are used:

- <<Geometry>> represents the geometry of an object. The geometry is an identifiable and distinguishable object that is derived from the abstract GML type *AbstractGeometryType*.
- <<Feature>> represents a thematic feature according to the definition in ISO 19109. A feature is an identifiable and distinguishable object that is derived from the abstract GML type *AbstractFeatureType*.
- <<Object>> represents an identifiable and distinguishable object that is derived from the abstract GML type *AbstractGMLType*.
- <<Enumeration>> enumerates the valid attribute values in a fixed list of named literal values. Enumerations are specified inline the CityGML schema.
- <<CodeList>> enumerates the valid attribute values. In contrast to Enumeration, the list of values is open and, thus, not given inline the CityGML schema. The allowed values can be provided within an external code list. It is recommended that code lists are implemented as simple dictionaries following the GML 3.1.1 Simple Dictionary Profile (cf. chapter 6.6 and chapter 10.14).
- <<Union>> is a list of attributes. The semantics are that only one of the attributes can be present at any time.
- <<PrimitiveType>> is used for representations supported by a primitive type in the implementation.
- <<DataType>> is used as a descriptor of a set of values that lack identity. Data types include primitive pre-defined types and user-definable types. A DataType is thus a class with few or no operations whose primary purpose is to hold the abstract state of another class for transmittal, storage, encoding or persistent storage.
- <<Leaf>> is used within UML package diagrams to indicate model elements that can have no further subtypes.
- <<XSDSchema>> is used within UML package diagrams to denote the root element of an XSD Schema containing all the definitions for a particular namespace. All the package contents or component classes are placed within the one schema.
- <<ApplicationSchema>> is used within UML package diagrams to denote an XML Schema definition fundamentally dependent on the concepts of another independent Standard within the XML Schema metalinguage. For example, ApplicationSchema indicates extensions of GML consistent with the GML “rules for application schemas”.

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

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

- Classes painted in blue belong to a CityGML UML package different to that associated with the yellow color. In order to explicitly denote the UML package of such classes, their class names carry a namespace prefix which is uniquely associated with a CityGML module throughout this specification (cf. section 4.3 for a list of namespaces and prefixes). For example, in the context of the *Building* module, classes from the *CityGML Core* module are painted in blue and their class names are preceded by the prefix *core*.
- Classes painted in green are defined in GML3 and their class names are preceded by the prefix *gml*.

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

Visual Paradigm for UML Standard Edition (Technische Universität Berlin)

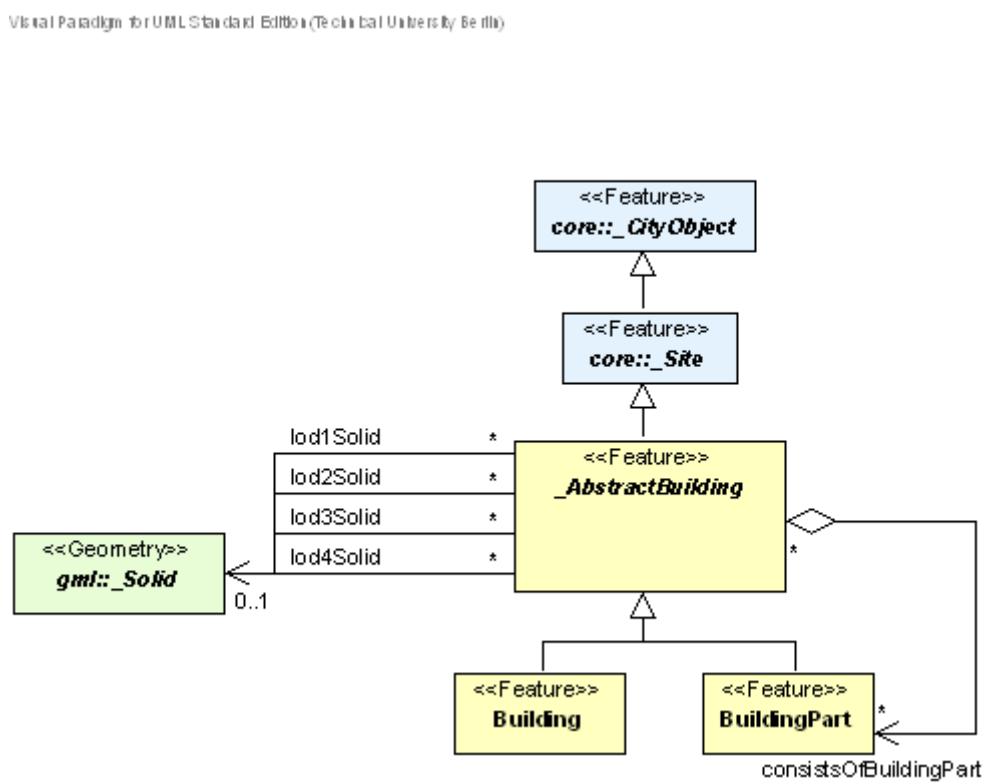


Figure 2. Example UML diagram demonstrating the UML notation and coloring scheme used throughout the CityGML specification.

## 6.3. XML namespaces and namespace prefixes

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

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

*Table 3. List of CityGML modules, their associated XML namespace identifiers, and example namespace prefixes.*

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

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

*Table 4. List of XML Schema definitions, their associated XML namespace identifiers, and example namespace prefixes used within this document.*

XML Schema Definition	Namespace identifier	Namespace prefix
Geography Markup Language version 3.1.1 (from OGC)	<a href="http://www.opengis.net/gml">http://www.opengis.net/gml</a>	gml
Extensible Address Language version 2.0 (from OASIS)	urn:oasis:names:tc:ciq:xsdschema: xAL:2.0	xAL
Schematron Assertion Lan-guage version 1.5	<a href="http://www.ascc.net/xml/&lt;br/&gt;schematron">http://www.ascc.net/xml/ schematron</a>	sch

**NOTE** Explanatory text edit 05 for 01.04.2020 release ends here

**NOTE** Explanatory text edit 06 for 01.04.2020 release starts here

# 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 international standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211.

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

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

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

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

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

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

**NOTE** Explanatory text edit 06 for 01.04.2020 release ends here

**NOTE** Explanatory text edit 07 for 01.04.2020 release starts here

# Chapter 8. General characteristics of CityGML

## 8.1. Modularisation

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

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

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

## 8.2. Multi-scale modelling (5 levels of detail, LOD)

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

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

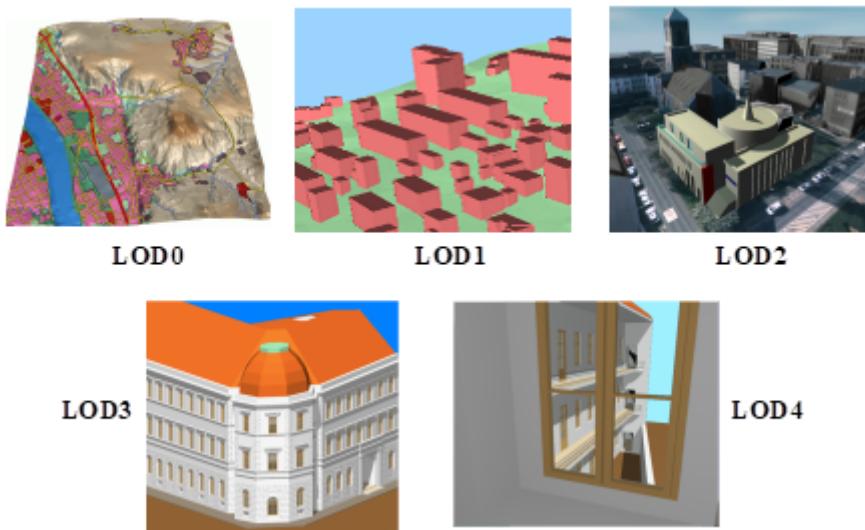


Figure 3. The five levels of detail (LOD) defined by CityGML (source: IGG Uni Bonn)

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

Table 5. LOD 0-4 of CityGML with their proposed accuracy requirements (discussion proposal, based on: Albert et al. 2003).

	<b>LOD0</b>	<b>LOD1</b>	<b>LOD2</b>	<b>LOD3</b>	<b>LOD4</b>
Model scale description	regional, landscape	city, region	city, city districts, projects	city districts, architectural models (exterior), landmark	architectural models (interior), landmark
Class of accuracy	lowest	low	middle	high	very high
Absolute 3D point accuracy (position / height)	lower than LOD1	5/5m	2/2m	0.5/0.5m	0.2/0.2m

	<b>LOD0</b>	<b>LOD1</b>	<b>LOD2</b>	<b>LOD3</b>	<b>LOD4</b>
Generalisation	maximal generalisation	object blocks as generalised features; > 6*6m/3m	objects as generalised features; > 4*4m/2m	object as real features; > 2*2m/1m	constructive elements and openings are represented
Building installations	no	no	yes	representative exterior features	real object form
Roof structure/representation	yes	flat	differentiated roof structures	real object form	real object form
Roof overhanging parts	yes	no	yes, if known	yes	yes
CityFurniture	no	important objects	prototypes, generalized objects	real object form	real object form
SolitaryVegetationObject	no	important objects	prototypes, higher 6m	prototypes, higher 2m	prototypes, real object form
PlantCover	no	>50*50m	>5*5m	< LOD2	<LOD2
...to be continued for the other feature themes					

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

## 8.3. Coherent semantical-geometrical modelling

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

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

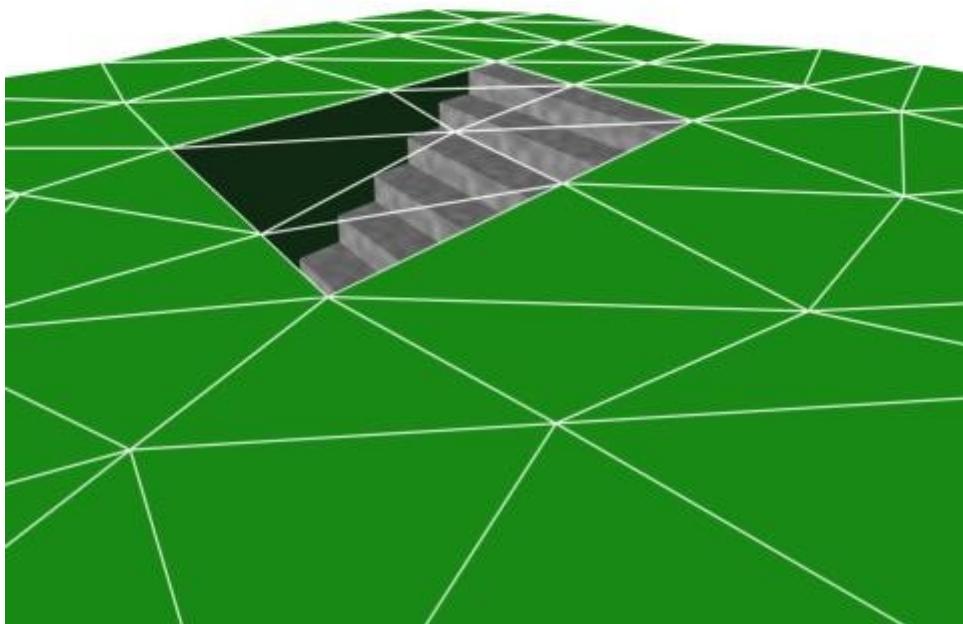
## 8.4. Closure surfaces

Objects, which are not modelled by a volumetric geometry, must be virtually closed in order to compute their volume (e.g. pedestrian underpasses or airplane hangars). They can be sealed using a ClosureSurface. These are special surfaces, which are taken into account, when needed to compute volumes and are neglected, when they are irrelevant or not appropriate, for example in visualisations.

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

**NOTE** Combine Figures 4





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

## 8.5. Terrain Intersection Curve (TIC)

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

the intersection with the terrain may differ depending on the LOD, a 3D object may have different TerrainIntersectionCurves for all LOD.

**NOTE** Combine figures 5





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

## 8.6. Code lists for enumerative attributes

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

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

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

list mechanism is provided in chapter 10.14.

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

## 8.7. External references

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

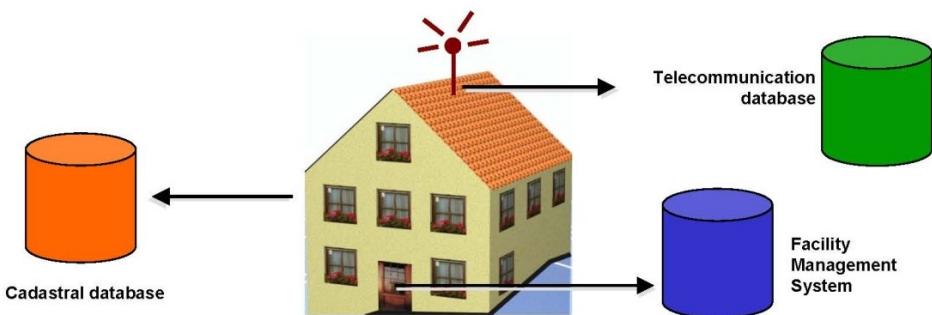


Figure 6. External references (graphic: IGG Uni Bonn).

## 8.8. City object groups

The grouping concept of CityGML allows for the aggregation of arbitrary city objects according to user-defined criteria, and to represent and transfer these aggregations as part of a city model (for the UML diagram see chapter 10.11; XML schema definition see annex A.6). A group may be assigned one or more names and may be further classified by specific attributes, for example, "escape route from room no. 43 in house no. 1212 in a fire scenario" as a name and "escape route" as type. Each member of the group can optionally be assigned a role name, which specifies the role this particular member plays in the group. This role name may, for example, describe the sequence number of this object in an escape route, or in the case of a building complex, denote the main building.

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

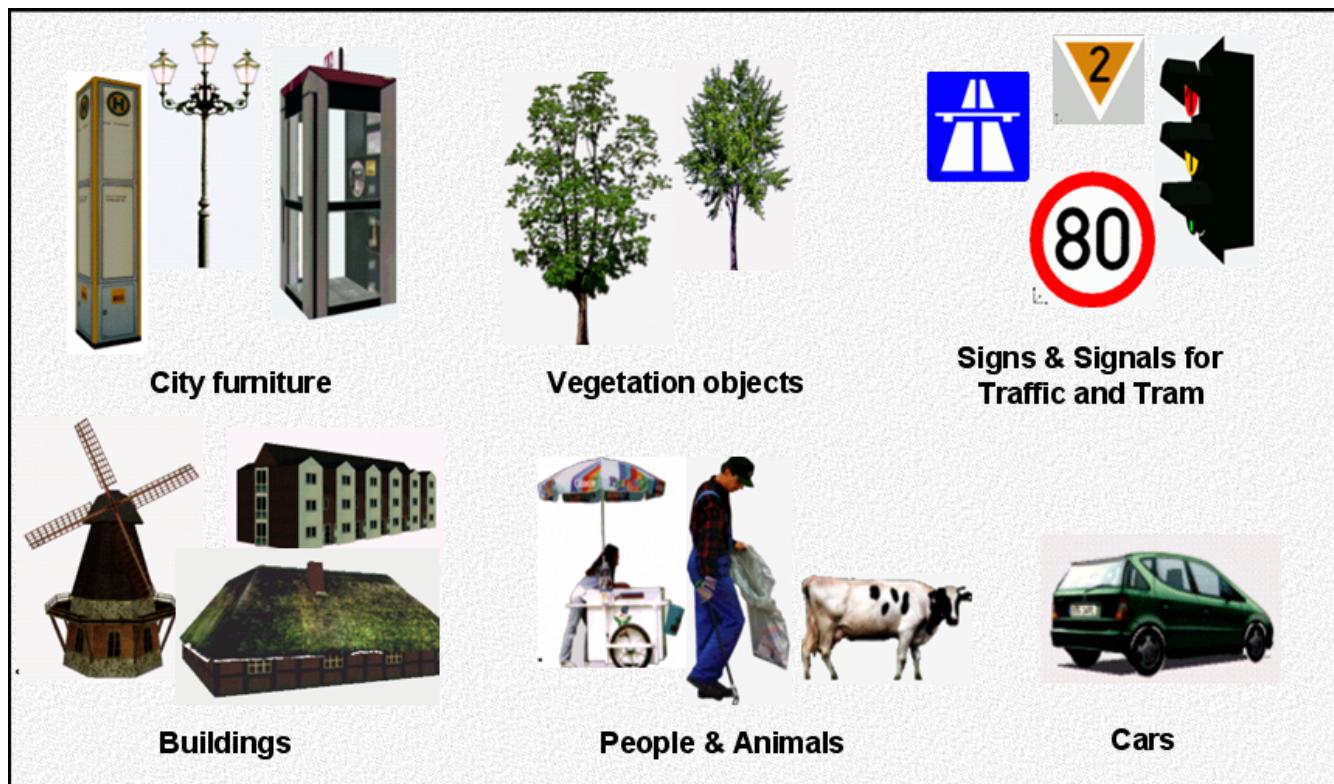
## 8.9. Appearances

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

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

## 8.10. Prototypic objects / scene graph concepts

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



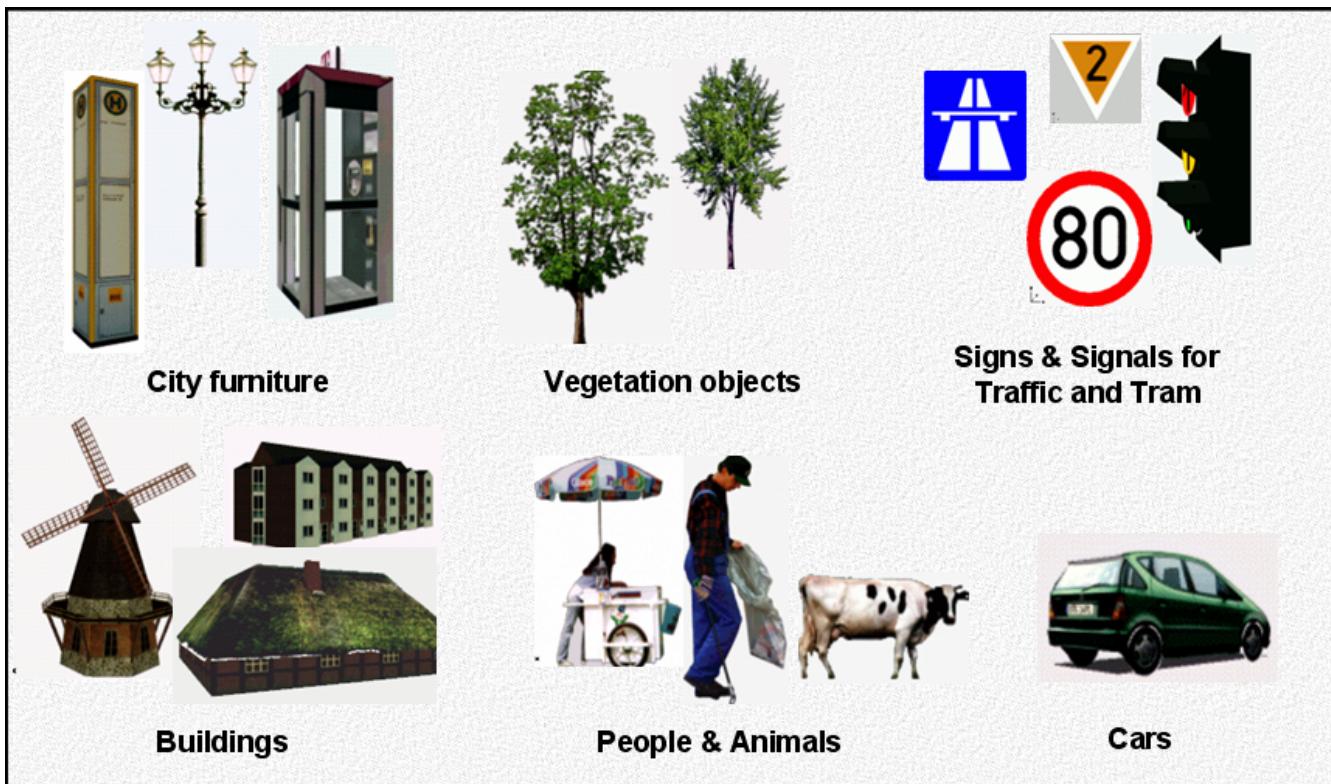


Figure 7. Examples of prototypic shapes (source: Rheinmetall Defence Electronics).

## 8.11. Generic city objects and attributes

CityGML is being designed as a universal topographic information model that defines object types and attributes which are useful for a broad range of applications. In practical applications the objects within specific 3D city models will most likely contain attributes which are not explicitly modelled in CityGML. Moreover, there might be 3D objects which are not covered by the thematic classes of CityGML. CityGML provides two different concepts to support the exchange of such data: 1) generic objects and attributes, and 2) Application Domain Extensions (cf. chapter 6.12).

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

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

## 8.12. Application Domain Extensions (ADE)

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

extended CityGML modules.

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

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

In this specification, two examples for ADEs are included:

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

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

**NOTE** Explanatory text edit 07 for 01.04.2020 release ends here

**NOTE** Explanatory text edit 08 for 01.04.2020 release starts here

# Chapter 9. Modularization

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

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

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

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

- Appearance (cf. clause 9),
- Bridge (cf. clause 10.5)
- Building (cf. clause 10.3),
- CityFurniture (cf. clause 10.9),
- CityObjectGroup (cf. clause 10.11),
- Generics (cf. clause 10.12),
- LandUse (cf. clause 10.10),

- Relief (cf. clause 10.2),
- Transportation (cf. clause 10.7),
- Tunnel (cf. clause 10.4)
- Vegetation (cf. clause 10.8),
- WaterBody (cf. clause 10.6), and
- TexturedSurface [deprecated] (cf. clause 9.8).

The thematic decomposition of the CityGML data model allows for implementations to support any combination of extension modules in conjunction with the core module in order to be CityGML conformant. Thus, the extension modules may be arbitrarily combined according to the information needs of an application or application domain. A combination of modules is called a CityGML profile. The union of all modules is defined as the CityGML base profile. The base profile is unique at any given time and forms the upper bound of the overall CityGML data model. Any other CityGML profile must be a valid subset of the base profile. By following the concept of CityGML modules and profiles, valid partial implementations of the CityGML data model may be realised in a well-defined way.

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

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

## 9.1. CityGML core and extension modules

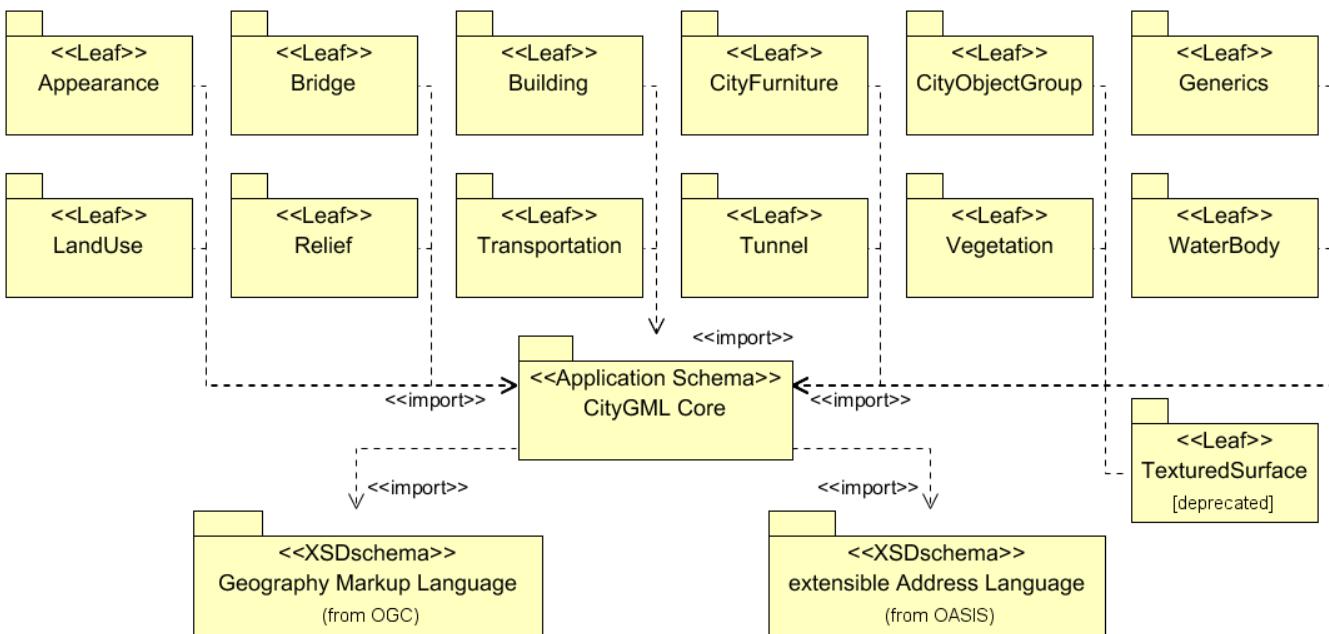
Each CityGML module is specified by its own XML Schema definition file and is defined within an individual and globally unique XML target namespace. According to dependency relations between modules, each module may, in addition, import namespaces associated to such related CityGML modules. However,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

module elements are guaranteed to be properly separated and distinguishable in CityGML instance documents.

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

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

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



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

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

Module name	Appearance
XML namespace identifier	<a href="http://www.opengis.net/citygml/appearance/2.0">http://www.opengis.net/citygml/appearance/2.0</a>
XML Schema file	appearance.xsd

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

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

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

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

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

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

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

Module name	Relief
-------------	--------

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

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

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

Module name	Vegetation
XML namespace identifier	<a href="http://www.opengis.net/citygml/vegetation/2.0">http://www.opengis.net/citygml/vegetation/2.0</a>
XML Schema file	vegetation.xsd
Recommended namespace prefix	veg

Module description	The Vegetation module provides thematic classes to represent vegetation objects. CityGML's vegetation model distinguishes between solitary vegetation objects like trees, and vegetation areas which represent biotopes like forests or other plant communities.
--------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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

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

## 9.2. CityGML profiles

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

1. CityGML profile definition embedded inline the CityGML instance document A CityGML profile can be bound to an instance document using the schemaLocation attribute defined in the XML Schema instance namespace, <http://www.w3.org/2001/XMLSchema-instance> (commonly associated with the prefix xsi). The xsi:schemaLocation attribute provides a way to locate the XML Schema definition for namespaces defined in an XML instance document. Its value is a

whitespace-delimited list of pairs of Uniform Resource Identifiers (URIs) where each pair consists of a namespace followed by the location of that namespace's XML Schema definition, which is typically a .xsd file.

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

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

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

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

**NOTE** replace XML with UML if feasible.

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

**NOTE** replace XML with UML if feasible

**NOTE** Explanatory text edit 08 for 01.04.2020 release ends here

**NOTE** Explanatory text edit 09 for 01.04.2020 release starts here

# Chapter 10. Spatial Model

**NOTE**

This section was not generated from the CityGML Conceptual Model. TODO: identify where this info resides in the UML model, then update accordingly.

Spatial properties of CityGML features are represented by objects of GML3's geometry model. This model is based on the standard ISO 19107 'Spatial Schema' (Herring 2001), representing 3D geometry according to the well-known Boundary Representation (B-Rep, cf. Foley et al. 1995).

CityGML actually uses only a subset of the GML3 geometry package, defining a profile of GML3. This subset is depicted in Fig. 9 and Fig. 10. Further-more, GML3's explicit Boundary Representation is extended by scene graph concepts, which allow the representation of the geometry of features with the same shape implicitly and thus more space efficiently (chapter 8.2).

**NOTE**

Version 2.0 only provides conformance requirements for implicit geometries. Additional requirements will be needed for the other categories.

## 10.1. Geometric-topological model

**NOTE**

short intro here

### 10.1.1. Primitives and Composites

**NOTE**

short intro here

For a more detailed discussion of Primitives and Composites, see [CityGML Best Practice Section nnn](#).

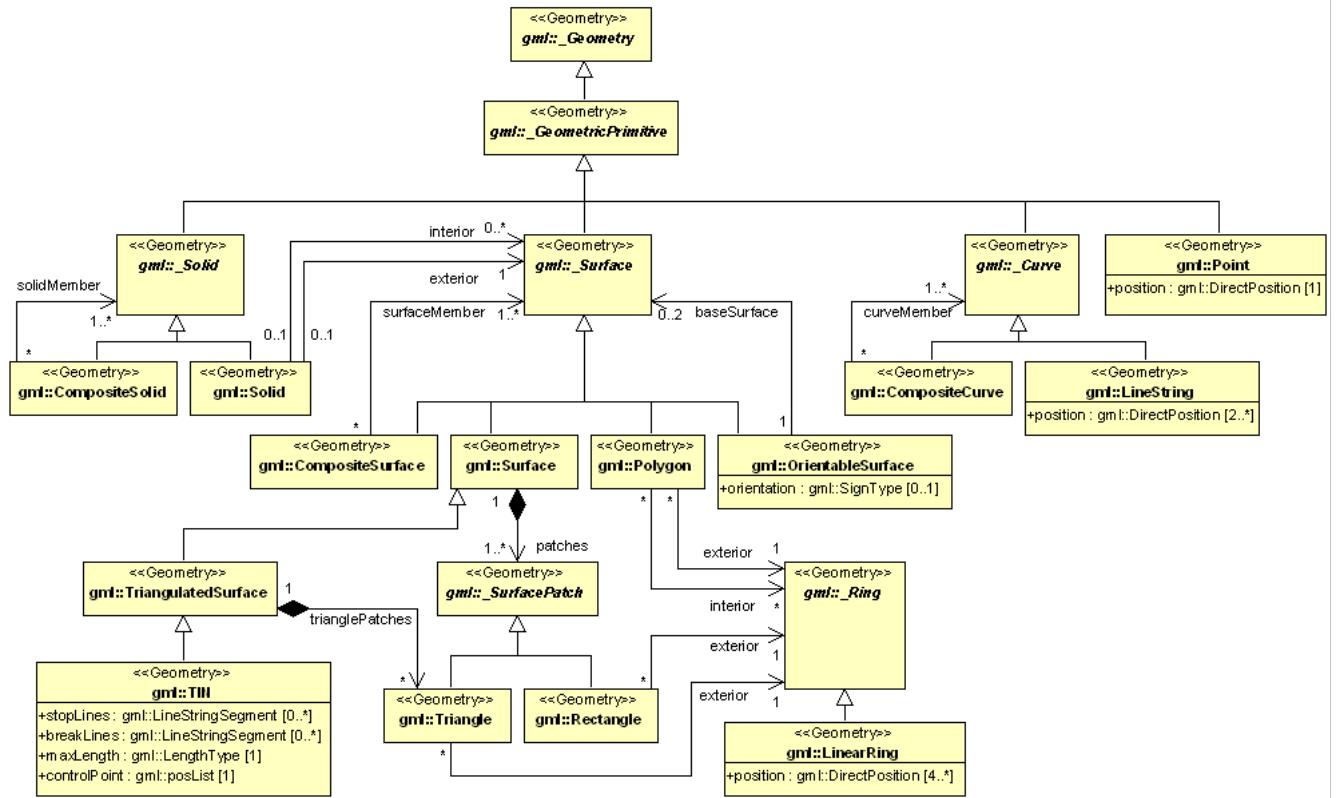


Figure 9. UML diagram of CityGML's geometry model (subset and profile of GML3): Primitives and Composites.

### 10.1.2. Complexes and Aggregates

**NOTE** short intro here

For a more detailed discussion of Complexes and Aggregates, see [CityGML Best Practice Section nnn](#).

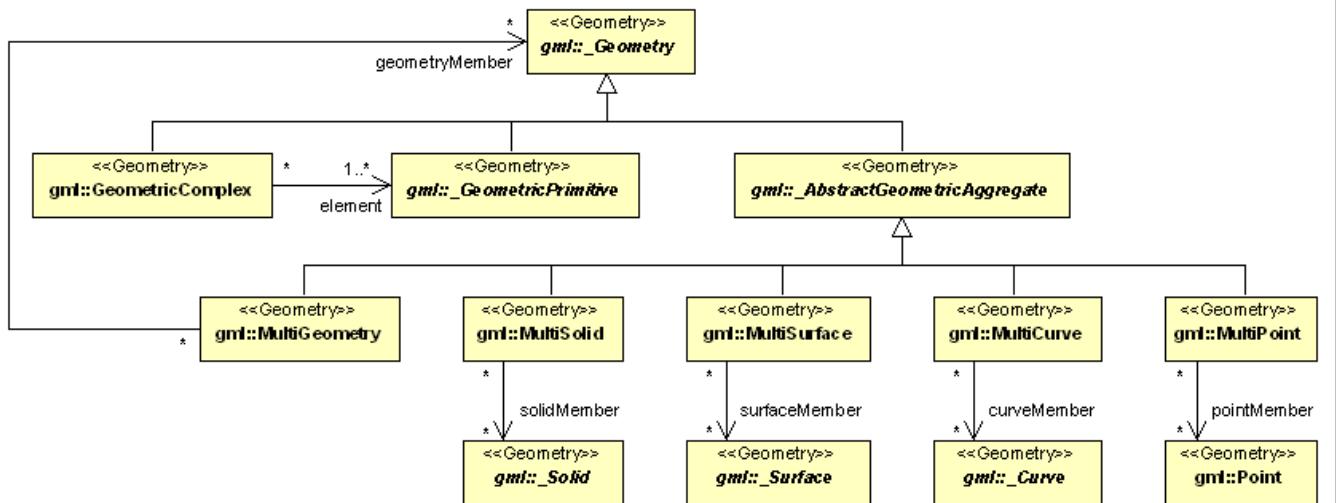


Figure 10. UML diagram of CityGML's geometry model: Complexes and Aggregates

### 10.1.3. Combined Geometries

**NOTE** | short intro here

For a more detailed discussion of Combined Geometries, see [CityGML Best Practice Section nnn](#).

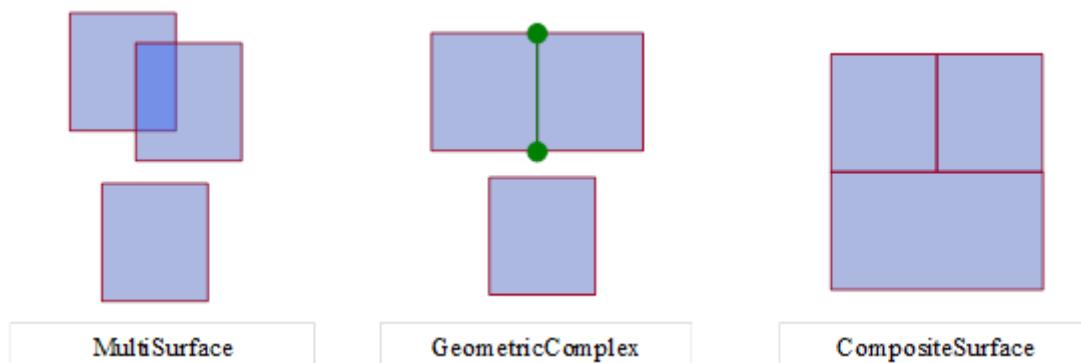


Figure 11. Combined geometries

#### 10.1.4. Recursive Aggregation

**NOTE** | short intro here

For a more detailed discussion of Recursive Aggregation, see [CityGML Best Practice Section nnn](#).

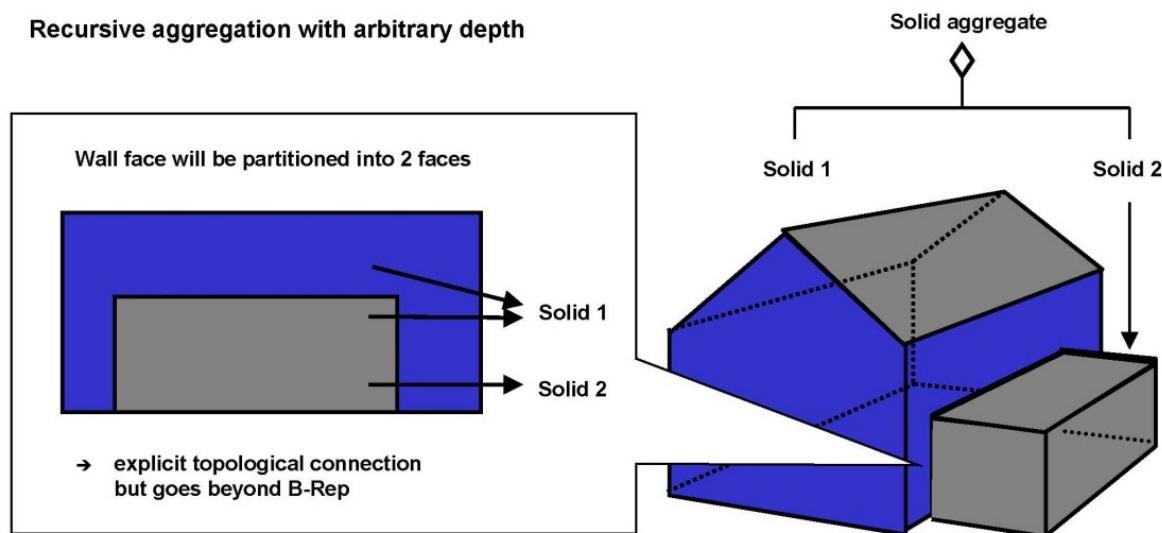


Figure 12. Recursive aggregation of objects and geometries in CityGML (graphic: IGG Uni Bonn).

## 10.2. Spatial Reference System

NOTE [short intro here](#)

For a more detailed discussion of Spatial Reference Systems, see [CityGML Best Practice Section nnn](#).

NOTE [add SRS requirements here](#)

## 10.3. Implicit geometries, prototypic objects, scene graph concepts

NOTE [short intro here](#)

For a more detailed discussion of Implicit Geometries, see [CityGML Best Practice Section nnn](#).

Visual Paradigm for UML Standard Edition (Technische Universität Berlin)

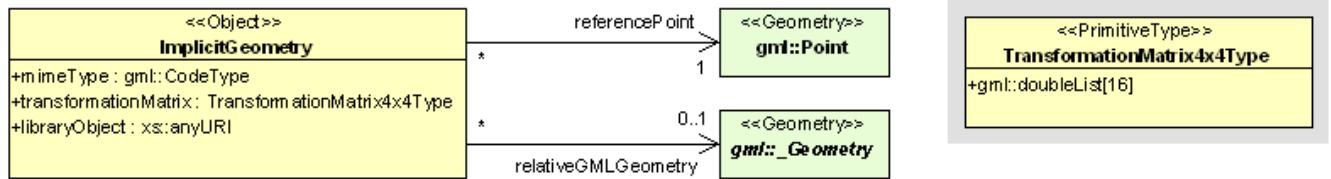


Figure 13. UML diagram of *ImplicitGeometries*. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Core module.

NOTE [Explanatory text edit 09 for 01.04.2020 release ends here](#)

# Chapter 11. CityGML Conceptual Model

This section provides a detailed discussion of the CityGML Conceptual Model.

## 11.1. Core

Requirements Class	
<a href="http://www.opengis.net/spec/CityGML/3.0/req/req-class-core">http://www.opengis.net/spec/CityGML/3.0/req/req-class-core</a>	
Target type	Conceptual Model
Dependency	None

The Core module defines the basic concepts and components of city models. 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.

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 areal extent in the real world. Space boundaries can be differentiated into different types of thematic surfaces, such as wall surfaces and roof surfaces. For a more detailed introduction to the Space concept see section [xxx](#).

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. For a more detailed introduction to the Level of Detail concept see section [xxx](#). 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.

City objects can have various relations. City objects can relate to generalized representations of the same real-world object in different Levels of Detail, they can also relate to other city objects, and they can reference corresponding external objects in other information systems, such as in the German cadastre (ALKIS), the German topographic information system (ATKIS), or the OS UK MasterMap®. In addition, address information can be specified for city objects using the xAL

address standard issued by the OASIS consortium (OASIS 2003).

The UML diagram of the Core module is depicted in [Core UML Diagram](#). A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

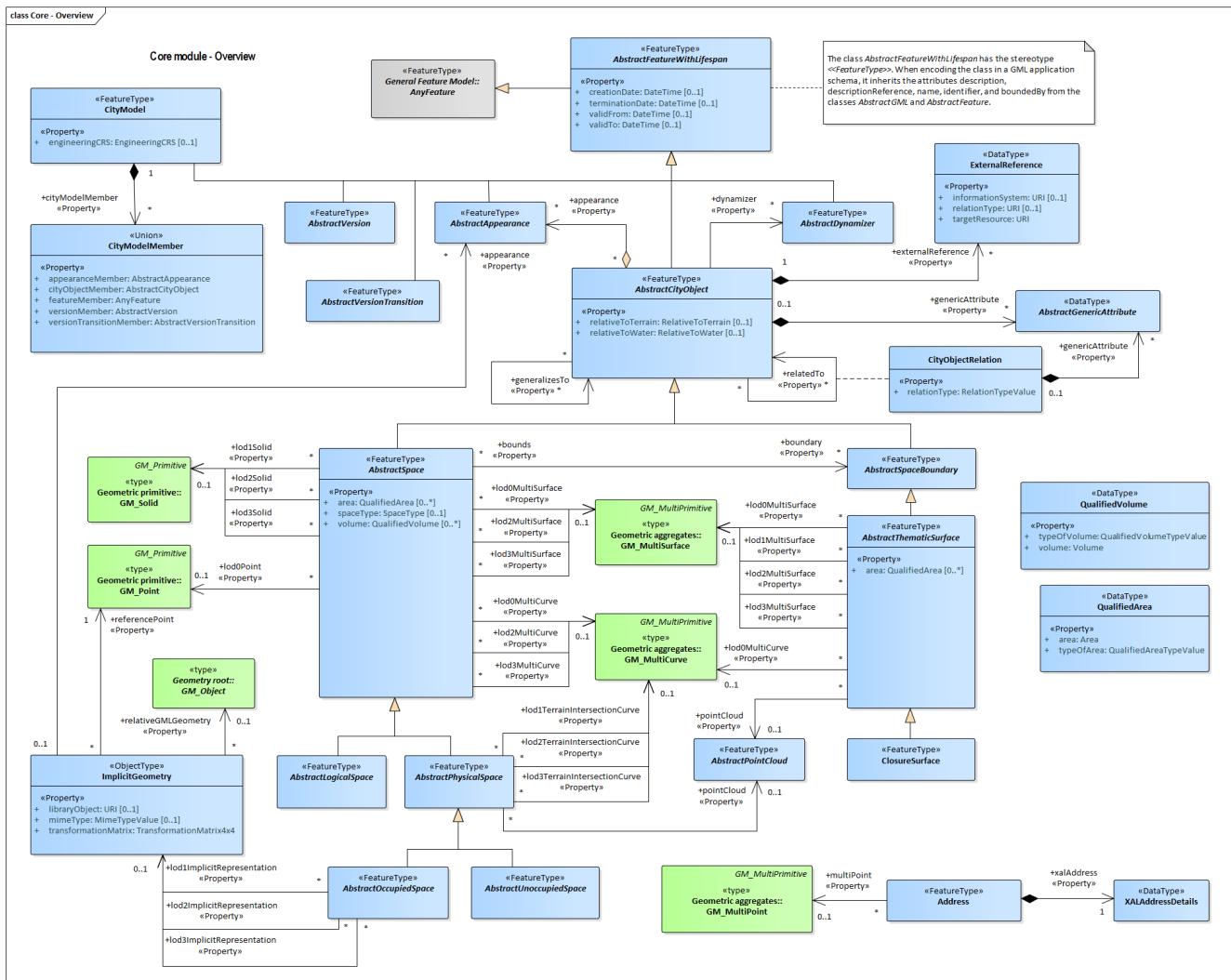


Figure 14. UML diagram of CityGML's core module.

### 11.1.1. Core 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.

**Stereotype:** «ApplicationSchema»

**Parent Package:** CityGML

### 11.1.2. Class AbstractAppearance

Requirement 1	/req/Core/AbstractAppearance
---------------	------------------------------

A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractAppearance UML class as documented in the AbstractAppearance section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractAppearance UML class as documented in the AbstractAppearance section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractAppearance UML class; including the name, definition, type, and cardinality of those documented in the AbstractAppearance section of the <a href="#">Core Data Dictionary</a> .

### Class AbstractAppearance

Definition: AbstractAppearance is the abstract superclass to represent any kind of appearance objects.

Subclass Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

### Relations

### Attributes

## 11.1.3. Class AbstractCityObject

Requirement 2	/req/Core/AbstractCityObject
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractCityObject UML class as documented in the AbstractCityObject section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractCityObject UML class as documented in the AbstractCityObject section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractCityObject UML class; including the name, definition, type, and cardinality of those documented in the AbstractCityObject section of the <a href="#">Core Data Dictionary</a> .

## **Class AbstractCityObject**

Definition: AbstractCityObject is the abstract superclass of all thematic classes within the CityGML data model.

Subclass Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

## **Relations**

Relation Name: externalReference

Cardinality: \*

Target Class: [ExternalReference](#)

Definition: References external objects in other information systems that have a relation to the city object.

Relation Name: dynamizer

Cardinality: \*

Target Class: [AbstractDynamizer](#)

Definition: Relates Dynamizer objects to the city object. These allow timeseries data to override static attribute values of the city object.

Relation Name: relatedTo

Cardinality: \*

Target Class: [AbstractCityObject](#)

Definition: Relates the city object to other city objects. It also describes how the city objects are related to each other.

Relation Name: appearance

Cardinality: \*

Target Class: [AbstractAppearance](#)

Definition: Relates appearances to the city object.

Relation Name: genericAttribute

Cardinality: \*

Target Class: [AbstractGenericAttribute](#)

Definition: Relates generic attributes to the city object.

Relation Name: generalizesTo

Cardinality: \*

Target Class: [AbstractCityObject](#)

Definition: Relates the city object to generalized representations of the same real-world object in different Levels of Detail. The direction of this relation is from the city object to the corresponding generalised city objects.

## **Attributes**

Attribute Name: relativeToTerrain

Value Type: RelativeToTerrain

Definition: Describes the vertical position of the city object relative to the surrounding terrain.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	relativeToWater
Value Type:	RelativeToWater
Definition:	Describes the vertical position of the city object relative to the surrounding water surface.
Multiplicity:	[0..1]
Stereotype:	«Property»

#### 11.1.4. Class AbstractDynamizer

Requirement 3	/req/Core/AbstractDynamizer
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractDynamizer UML class as documented in the AbstractDynamizer section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractDynamizer UML class as documented in the AbstractDynamizer section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractDynamizer UML class; including the name, definition, type, and cardinality of those documented in the AbstractDynamizer section of the <a href="#">Core Data Dictionary</a> .

#### Class AbstractDynamizer

Definition:	AbstractDynamizer is the abstract superclass to represent Dynamizer objects.
Subclass Of:	<a href="#">AbstractFeatureWithLifespan</a>
Stereotype:	«FeatureType»

#### Relations

#### Attributes

#### 11.1.5. Class AbstractFeatureWithLifespan

Requirement 4	/req/Core/AbstractFeatureWithLifespan
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractFeatureWithLifespan UML class as documented in the AbstractFeatureWithLifespan section of the <a href="#">Core Data Dictionary</a> .

B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractFeatureWithLifespan UML class as documented in the AbstractFeatureWithLifespan section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractFeatureWithLifespan UML class; including the name, definition, type, and cardinality of those documented in the AbstractFeatureWithLifespan section of the <a href="#">Core Data Dictionary</a> .

### Class 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: [AnyFeature](#)

Stereotype: «FeatureType»

### Relations

### Attributes

Attribute Name: creationDate

Value Type: DateTime

Definition: Indicates the date at which a CityGML feature was added to the CityModel.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: terminationDate

Value Type: DateTime

Definition: Indicates the date at which a CityGML feature was removed from the CityModel.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: validFrom

Value Type: DateTime

Definition: Indicates the date at which a CityGML feature started to exist in the real world.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: validTo

Value Type: DateTime

Definition: Indicates the date at which a CityGML feature ended to exist in the real world.

Multiplicity: [0..1]

Stereotype: «Property»

## 11.1.6. Class AbstractLogicalSpace

Requirement 5	/req/Core/AbstractLogicalSpace
---------------	--------------------------------

A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractLogicalSpace UML class as documented in the AbstractLogicalSpace section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractLogicalSpace UML class as documented in the AbstractLogicalSpace section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractLogicalSpace UML class; including the name, definition, type, and cardinality of those documented in the AbstractLogicalSpace section of the <a href="#">Core Data Dictionary</a> .

### **Class 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»

#### **Relations**

#### **Attributes**

### **11.1.7. Class AbstractOccupiedSpace**

Requirement 6	/req/Core/AbstractOccupiedSpace
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractOccupiedSpace UML class as documented in the AbstractOccupiedSpace section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractOccupiedSpace UML class as documented in the AbstractOccupiedSpace section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractOccupiedSpace UML class; including the name, definition, type, and cardinality of those documented in the AbstractOccupiedSpace section of the <a href="#">Core Data Dictionary</a> .

## Class 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»

## Relations

Relation Name: lod1ImplicitRepresentation

Cardinality: 0..1

Target Class: [ImplicitGeometry](#)

Definition: Relates an implicit geometry that represents the occupied space in Level of Detail 1.

Relation Name: lod2ImplicitRepresentation

Cardinality: 0..1

Target Class: [ImplicitGeometry](#)

Definition: Relates an implicit geometry that represents the occupied space in Level of Detail 2.

Relation Name: lod3ImplicitRepresentation

Cardinality: 0..1

Target Class: [ImplicitGeometry](#)

Definition: Relates an implicit geometry that represents the occupied space in Level of Detail 3.

## Attributes

### 11.1.8. Class AbstractPhysicalSpace

Requirement 7	/req/Core/AbstractPhysicalSpace
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractPhysicalSpace UML class as documented in the AbstractPhysicalSpace section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractPhysicalSpace UML class as documented in the AbstractPhysicalSpace section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractPhysicalSpace UML class; including the name, definition, type, and cardinality of those documented in the AbstractPhysicalSpace section of the <a href="#">Core Data Dictionary</a> .

## **Class 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»

## **Relations**

Relation Name: lod3TerrainIntersectionCurve

Cardinality: 0..1

Target Class: [GM\\_MultiCurve](#)

Definition: Relates a 3D MultiCurve geometry that represents the terrain intersection curve of the physical space in Level of Detail 3.

Relation Name: lod2TerrainIntersectionCurve

Cardinality: 0..1

Target Class: [GM\\_MultiCurve](#)

Definition: Relates a 3D MultiCurve geometry that represents the terrain intersection curve of the physical space in Level of Detail 2.

Relation Name: lod1TerrainIntersectionCurve

Cardinality: 0..1

Target Class: [GM\\_MultiCurve](#)

Definition: Relates a 3D MultiCurve geometry that represents the terrain intersection curve of the physical space in Level of Detail 1.

Relation Name: pointCloud

Cardinality: 0..1

Target Class: [AbstractPointCloud](#)

Definition: Relates a 3D PointCloud that represents the physical space.

## **Attributes**

### **11.1.9. Class AbstractPointCloud**

<b>Requirement 8</b>	<a href="#">/req/Core/AbstractPointCloud</a>
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractPointCloud UML class as documented in the AbstractPointCloud section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractPointCloud UML class as documented in the AbstractPointCloud section of the <a href="#">Core Data Dictionary</a> .

C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractPointCloud UML class; including the name, definition, type, and cardinality of those documented in the AbstractPointCloud section of the <a href="#">Core Data Dictionary</a> .
---	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class AbstractPointCloud

Definition: AbstractPointCloud is the abstract superclass to represent PointCloud objects.

Subclass Of: <– section,>>

Stereotype: «FeatureType»

#### Relations

#### Attributes

### 11.1.10. Class AbstractSpace

Requirement 9	/req/Core/AbstractSpace
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractSpace UML class as documented in the AbstractSpace section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractSpace UML class as documented in the AbstractSpace section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractSpace UML class; including the name, definition, type, and cardinality of those documented in the AbstractSpace section of the <a href="#">Core Data Dictionary</a> .

### Class 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»

#### Relations

Relation Name: boundary

Cardinality: \*

Target Class: [AbstractSpaceBoundary](#)

Definition: Relates surfaces that bound the space.

Relation Name:	lod2MultiCurve
Cardinality:	0..1
Target Class:	<a href="#">GM_MultiCurve</a>
Definition:	Relates a 3D MultiCurve geometry that represents the space in Level of Detail 2.
Relation Name:	lod2MultiSurface
Cardinality:	0..1
Target Class:	<a href="#">GM_MultiSurface</a>
Definition:	Relates a 3D MultiSurface geometry that represents the space in Level of Detail 2.
Relation Name:	lod1Solid
Cardinality:	0..1
Target Class:	<a href="#">GM_Solid</a>
Definition:	Relates a 3D Solid geometry that represents the space in Level of Detail 1.
Relation Name:	lod3MultiSurface
Cardinality:	0..1
Target Class:	<a href="#">GM_MultiSurface</a>
Definition:	Relates a 3D MultiSurface geometry that represents the space in Level of Detail 3.
Relation Name:	lod2Solid
Cardinality:	0..1
Target Class:	<a href="#">GM_Solid</a>
Definition:	Relates a 3D Solid geometry that represents the space in Level of Detail 2.
Relation Name:	lod0Point
Cardinality:	0..1
Target Class:	<a href="#">GM_Point</a>
Definition:	Relates a 3D Point geometry that represents the space in Level of Detail 0.
Relation Name:	lod0MultiCurve
Cardinality:	0..1
Target Class:	<a href="#">GM_MultiCurve</a>
Definition:	Relates a 3D MultiCurve geometry that represents the space in Level of Detail 0.
Relation Name:	lod3MultiCurve
Cardinality:	0..1
Target Class:	<a href="#">GM_MultiCurve</a>
Definition:	Relates a 3D MultiCurve geometry that represents the space in Level of Detail 3.
Relation Name:	lod3Solid
Cardinality:	0..1
Target Class:	<a href="#">GM_Solid</a>
Definition:	Relates a 3D Solid geometry that represents the space in Level of Detail 3.
Relation Name:	lod0MultiSurface
Cardinality:	0..1
Target Class:	<a href="#">GM_MultiSurface</a>
Definition:	Relates a 3D MultiSurface geometry that represents the space in Level of Detail 0.
<b>Attributes</b>	

Attribute Name:	area
Value Type:	QualifiedArea
Definition:	Specifies qualified areas related to the space.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	spaceType
Value Type:	SpaceType
Definition:	Specifies the degree of openness of a space.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	volume
Value Type:	QualifiedVolume
Definition:	Specifies qualified volumes related to the space.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.1.11. Class AbstractSpaceBoundary

Requirement 10	/req/Core/AbstractSpaceBoundary
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractSpaceBoundary UML class as documented in the AbstractSpaceBoundary section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractSpaceBoundary UML class as documented in the AbstractSpaceBoundary section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractSpaceBoundary UML class; including the name, definition, type, and cardinality of those documented in the AbstractSpaceBoundary section of the <a href="#">Core Data Dictionary</a> .

#### Class 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»

#### Relations

#### Attributes

## 11.1.12. Class AbstractThematicSurface

Requirement 11	/req/Core/AbstractThematicSurface
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractThematicSurface UML class as documented in the AbstractThematicSurface section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractThematicSurface UML class as documented in the AbstractThematicSurface section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractThematicSurface UML class; including the name, definition, type, and cardinality of those documented in the AbstractThematicSurface section of the <a href="#">Core Data Dictionary</a> .

### Class AbstractThematicSurface

Definition: AbstractThematicSurface is the abstract superclass for all types of thematic surfaces.

Subclass Of: [AbstractSpaceBoundary](#)

StereoType: «FeatureType»

### Relations

Relation Name: lod3MultiSurface

Cardinality: 0..1

Target Class: [GM\\_MultiSurface](#)

Definition: Relates a 3D MultiSurface geometry that represents the thematic surface in Level of Detail 3.

Relation Name: pointCloud

Cardinality: 0..1

Target Class: [AbstractPointCloud](#)

Definition: Relates a 3D PointCloud that represents the thematic surface.

Relation Name: lod0MultiCurve

Cardinality: 0..1

Target Class: [GM\\_MultiCurve](#)

Definition: Relates a 3D MultiCurve geometry that represents the thematic surface in Level of Detail 0.

Relation Name: lod1MultiSurface

Cardinality: 0..1

Target Class: [GM\\_MultiSurface](#)

Definition: Relates a 3D MultiSurface geometry that represents the thematic surface in Level of Detail 1.

Relation Name:	lod2MultiSurface
Cardinality:	0..1
Target Class:	<a href="#">GM_MultiSurface</a>
Definition:	Relates a 3D MultiSurface geometry that represents the thematic surface in Level of Detail 2.
Relation Name:	lod0MultiSurface
Cardinality:	0..1
Target Class:	<a href="#">GM_MultiSurface</a>
Definition:	Relates a 3D MultiSurface geometry that represents the thematic surface in Level of Detail 0.
<b>Attributes</b>	
Attribute Name:	area
Value Type:	<a href="#">QualifiedArea</a>
Definition:	Specifies qualified areas related to the thematic surface.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.1.13. Class AbstractUnoccupiedSpace

Requirement 12	/req/Core/AbstractUnoccupiedSpace
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractUnoccupiedSpace UML class as documented in the AbstractUnoccupiedSpace section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractUnoccupiedSpace UML class as documented in the AbstractUnoccupiedSpace section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractUnoccupiedSpace UML class; including the name, definition, type, and cardinality of those documented in the AbstractUnoccupiedSpace section of the <a href="#">Core Data Dictionary</a> .

#### Class 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»

#### Relations

#### Attributes

## 11.1.14. Class AbstractVersion

Requirement 13	/req/Core/AbstractVersion
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractVersion UML class as documented in the AbstractVersion section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractVersion UML class as documented in the AbstractVersion section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractVersion UML class; including the name, definition, type, and cardinality of those documented in the AbstractVersion section of the <a href="#">Core Data Dictionary</a> .

### Class AbstractVersion

Definition: AbstractVersion is the abstract superclass to represent Version objects.

Subclass Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

### Relations

### Attributes

## 11.1.15. Class AbstractVersionTransition

Requirement 14	/req/Core/AbstractVersionTransition
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractVersionTransition UML class as documented in the AbstractVersionTransition section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractVersionTransition UML class as documented in the AbstractVersionTransition section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractVersionTransition UML class; including the name, definition, type, and cardinality of those documented in the AbstractVersionTransition section of the <a href="#">Core Data Dictionary</a> .

### **Class AbstractVersionTransition**

Definition: AbstractVersionTransition is the abstract superclass to represent VersionTransition objects.

Subclass Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

### **Relations**

### **Attributes**

## **11.1.16. Class Address**

<b>Requirement 15</b>	<b>/req/Core/Address</b>
A	The Implementation Specification SHALL contain an element with the same definition as the Address UML class as documented in the Address section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Address UML class as documented in the Address section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Address UML class; including the name, definition, type, and cardinality of those documented in the Address section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Address UML class; including the name, definition, type, and cardinality of those documented in the Address section of the <a href="#">Core Data Dictionary</a> .

### **Class Address**

Definition: Address represents an address of a city object.

Subclass Of: <-->

Stereotype: «FeatureType»

### **Relations**

Relation Name: xalAddress

Cardinality: 1

Target Class: [XALAddressDetails](#)

Definition: Relates to an OASIS address object.

Relation Name: multiPoint

Cardinality: 0..1

Target Class: [GM\\_MultiPoint](#)

Definition: Relates to the MultiPoint geometry of the Address. The geometry relates the address spatially to a city object.

### **Attributes**

## 11.1.17. Class CityModel

Requirement 16	/req/Core/CityModel
A	The Implementation Specification SHALL contain an element with the same definition as the CityModel UML class as documented in the CityModel section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the CityModel UML class as documented in the CityModel section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the CityModel UML class; including the name, definition, type, and cardinality of those documented in the CityModel section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the CityModel UML class; including the name, definition, type, and cardinality of those documented in the CityModel section of the <a href="#">Core Data Dictionary</a> .

### Class CityModel

Definition: CityModel is the container for all objects belonging to a city model.

Subclass Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

### Relations

Relation Name: cityModelMember

Cardinality: \*

Target Class: [CityModelMember](#)

Definition: References all objects that are part of the CityModel.

### Attributes

Attribute Name: engineeringCRS

Value Type: EngineeringCRS

Definition: Specifies the local coordinate reference system of the CityModel.

Multiplicity: [0..1]

Stereotype: «Property»

## 11.1.18. Class CityObjectRelation

Requirement 17	/req/Core/CityObjectRelation
A	Any use of the CityObjectRelation type in the Implementation Specification SHALL have the same definition as the CityObjectRelation UML class as documented in the CityObjectRelation section of the <a href="#">Core Data Dictionary</a> .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the CityObjectRelation UML class as documented in the CityObjectRelation section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the CityObjectRelation UML class; including the name, definition, type, and cardinality of those documented in the CityObjectRelation section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the CityObjectRelation UML class; including the name, definition, type, and cardinality of those documented in the CityObjectRelation section of the <a href="#">Core Data Dictionary</a> .

### Class CityObjectRelation

Definition: CityObjectRelation represents a specific relation from the city object in which it is included to another city object.

Subclass Of: <-- section,>>

Stereotype:

#### Relations

Relation Name: genericAttribute

Cardinality: \*

Target Class: [AbstractGenericAttribute](#)

Definition: Relates generic attributes to the CityObjectRelation.

#### Attributes

Attribute Name: relationType

Value Type: RelationTypeValue

Definition: Indicates the specific type of the CityObjectRelation.

Multiplicity:

Stereotype: «Property»

### 11.1.19. Class ClosureSurface

Requirement 18	/req/Core/ClosureSurface
A	The Implementation Specification SHALL contain an element with the same definition as the ClosureSurface UML class as documented in the ClosureSurface section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ClosureSurface UML class as documented in the ClosureSurface section of the <a href="#">Core Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the ClosureSurface UML class; including the name, definition, type, and cardinality of those documented in the ClosureSurface section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ClosureSurface UML class; including the name, definition, type, and cardinality of those documented in the ClosureSurface section of the <a href="#">Core Data Dictionary</a> .

### Class 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»

#### Relations

#### Attributes

### 11.1.20. Class DoubleBetween0and1

Requirement 19	/req/Core/DoubleBetween0and1
A	Any use of the DoubleBetween0and1 type in the Implementation Specification SHALL have the same definition as the DoubleBetween0and1 UML class as documented in the DoubleBetween0and1 section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the DoubleBetween0and1 UML class as documented in the DoubleBetween0and1 section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the DoubleBetween0and1 UML class; including the name, definition, type, and cardinality of those documented in the DoubleBetween0and1 section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the DoubleBetween0and1 UML class; including the name, definition, type, and cardinality of those documented in the DoubleBetween0and1 section of the <a href="#">Core Data Dictionary</a> .

## **Class 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: <-- section,>>

Stereotype: «BasicType»

### **Relations**

### **Attributes**

## **11.1.21. Class DoubleBetween0and1List**

Requirement 20	/req/Core/DoubleBetween0and1List
A	Any use of the DoubleBetween0and1List type in the Implementation Specification SHALL have the same definition as the DoubleBetween0and1List UML class as documented in the DoubleBetween0and1List section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the DoubleBetween0and1List UML class as documented in the DoubleBetween0and1List section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the DoubleBetween0and1List UML class; including the name, definition, type, and cardinality of those documented in the DoubleBetween0and1List section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the DoubleBetween0and1List UML class; including the name, definition, type, and cardinality of those documented in the DoubleBetween0and1List section of the <a href="#">Core Data Dictionary</a> .

## **Class 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: <-- section,>>

Stereotype: «BasicType»

### **Relations**

### **Attributes**

Attribute Name: list

Value Type: DoubleBetween0and1

Definition: Specifies the list of double values.

Multiplicity:

Stereotype:

## 11.1.22. Class DoubleList

Requirement 21	/req/Core/DoubleList
A	Any use of the DoubleList type in the Implementation Specification SHALL have the same definition as the DoubleList UML class as documented in the DoubleList section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the DoubleList UML class as documented in the DoubleList section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the DoubleList UML class; including the name, definition, type, and cardinality of those documented in the DoubleList section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the DoubleList UML class; including the name, definition, type, and cardinality of those documented in the DoubleList section of the <a href="#">Core Data Dictionary</a> .

### Class DoubleList

Definition: DoubleList is an ordered sequence of double values.

Subclass Of: <-->

Stereotype: «BasicType»

### Relations

### Attributes

Attribute Name: list

Value Type: Real

Definition: Specifies the list of double values.

Multiplicity:

Stereotype:

## 11.1.23. Class ImplicitGeometry

Requirement 22	/req/Core/ImplicitGeometry
A	The Implementation Specification SHALL contain an element with the same definition as the ImplicitGeometry UML class as documented in the ImplicitGeometry section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ImplicitGeometry UML class as documented in the ImplicitGeometry section of the <a href="#">Core Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the ImplicitGeometry UML class; including the name, definition, type, and cardinality of those documented in the ImplicitGeometry section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ImplicitGeometry UML class; including the name, definition, type, and cardinality of those documented in the ImplicitGeometry section of the <a href="#">Core Data Dictionary</a> .

## Class 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: <-- section,>>

Stereotype: «ObjectType»

## Relations

Relation Name: referencePoint

Cardinality: 1

Target Class: [GM\\_Point](#)

Definition: Relates to a 3D Point geometry that represents the base point of the object in the world coordinate system.

Relation Name: appearance

Cardinality: \*

Target Class: [AbstractAppearance](#)

Definition: Relates appearances to the ImplicitGeometry.

Relation Name: relativeGMLGeometry

Cardinality: 0..1

Target Class: [GM\\_Object](#)

Definition: Relates to a prototypical geometry in a local coordinate system stored inline with the city model.

## Attributes

Attribute Name: libraryObject

Value Type: URI

Definition: Specifies the URI that points to the prototypical geometry stored in an external file.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	mimeType
Value Type:	MimeTypeValue
Definition:	Specifies the MIME type of the external file that stores the prototypical geometry.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	transformationMatrix
Value Type:	TransformationMatrix4x4
Definition:	Specifies the mathematical transformation (translation, rotation, and scaling) between the prototypical geometry and the actual spatial position of the object.
Multiplicity:	
Stereotype:	«Property»

### 11.1.24. Class IntegerBetween0and3

Requirement 23	/req/Core/IntegerBetween0and3
A	Any use of the IntegerBetween0and3 type in the Implementation Specification SHALL have the same definition as the IntegerBetween0and3 UML class as documented in the IntegerBetween0and3 section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the IntegerBetween0and3 UML class as documented in the IntegerBetween0and3 section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the IntegerBetween0and3 UML class; including the name, definition, type, and cardinality of those documented in the IntegerBetween0and3 section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the IntegerBetween0and3 UML class; including the name, definition, type, and cardinality of those documented in the IntegerBetween0and3 section of the <a href="#">Core Data Dictionary</a> .

#### Class 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: <-> section,>>

Stereotype: «BasicType»

#### Relations

#### Attributes

## 11.1.25. Class IntervalValue

Requirement 24	/req/Core/IntervalValue
A	Any use of the IntervalValue type in an Implementation Specification SHALL be restricted to the valid values specified in the IntervalValue UML class as documented in the IntervalValue section of the <a href="#">Core Data Dictionary</a> .

### Class IntervalValue

Definition: IntervalValue is a code list used to specify a time period.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.1.26. Class MimeJsonValue

Requirement 25	/req/Core/MimeJsonValue
A	Any use of the MimeJsonValue type in an Implementation Specification SHALL be restricted to the valid values specified in the MimeJsonValue UML class as documented in the MimeJsonValue section of the <a href="#">Core Data Dictionary</a> .

### Class MimeJsonValue

Definition: MimeJsonValue is a code list used to specify the MIME type of a referenced resource.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.1.27. Class OccupantTypeValue

Requirement 26	/req/Core/OccupantTypeValue
A	Any use of the OccupantTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the OccupantTypeValue UML class as documented in the OccupantTypeValue section of the <a href="#">Core Data Dictionary</a> .

### Class OccupantTypeValue

Definition: OccupantTypeValue is a code list used to classify occupants.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

**Relations****Attributes****11.1.28. Class OtherRelationTypeValue**

<b>Requirement 27</b>	/req/Core/OtherRelationTypeValue
A	Any use of the OtherRelationTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the OtherRelationTypeValue UML class as documented in the OtherRelationTypeValue section of the <a href="#">Core Data Dictionary</a> .

**Class OtherRelationTypeValue**

Definition: OtherRelationTypeValue is a code list used to classify other types of city object relations.

Subclass Of: [RelationTypeValue](#)

Stereotype: «CodeList»

**Relations****Attributes****11.1.29. Class QualifiedAreaTypeValue**

<b>Requirement 28</b>	/req/Core/QualifiedAreaTypeValue
A	Any use of the QualifiedAreaTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the QualifiedAreaTypeValue UML class as documented in the QualifiedAreaTypeValue section of the <a href="#">Core Data Dictionary</a> .

**Class QualifiedAreaTypeValue**

Definition: QualifiedAreaTypeValue is a code list used to specify area types.

Subclass Of: <-->

Stereotype: «CodeList»

**Relations****Attributes****11.1.30. Class QualifiedVolumeTypeValue**

<b>Requirement 29</b>	/req/Core/QualifiedVolumeTypeValue
-----------------------	------------------------------------

A	Any use of the QualifiedVolumeTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the QualifiedVolumeTypeValue UML class as documented in the QualifiedVolumeTypeValue section of the <a href="#">Core Data Dictionary</a> .
---	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class QualifiedVolumeTypeValue

Definition: QualifiedVolumeTypeValue is a code list used to specify volume types.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.1.31. Class RelationTypeValue

Requirement 30	/req/Core/RelationTypeValue
A	Any use of the RelationTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RelationTypeValue UML class as documented in the RelationTypeValue section of the <a href="#">Core Data Dictionary</a> .

### Class RelationTypeValue

Definition: RelationTypeValue is a code list used to classify city object relations.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.1.32. Class TemporalRelationTypeValue

Requirement 31	/req/Core/TemporalRelationTypeValue
A	Any use of the TemporalRelationTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TemporalRelationTypeValue UML class as documented in the TemporalRelationTypeValue section of the <a href="#">Core Data Dictionary</a> .

### Class TemporalRelationTypeValue

Definition: TemporalRelationTypeValue is a code list used to classify temporal city object relations.

Subclass Of: [RelationTypeValue](#)

Stereotype: «CodeList»

#### Relations

## Attributes

### 11.1.33. Class TopologicRelationTypeValue

Requirement 32	/req/Core/TopologicRelationTypeValue
A	Any use of the TopologicRelationTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TopologicRelationTypeValue UML class as documented in the TopologicRelationTypeValue section of the <a href="#">Core Data Dictionary</a> .

#### Class TopologicRelationTypeValue

Definition: TopologicRelationTypeValue is a code list used to classify topological city object relations.

Subclass Of: [RelationTypeValue](#)

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.1.34. Class TransformationMatrix2x2

Requirement 33	/req/Core/TransformationMatrix2x2
A	Any use of the TransformationMatrix2x2 type in the Implementation Specification SHALL have the same definition as the TransformationMatrix2x2 UML class as documented in the TransformationMatrix2x2 section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TransformationMatrix2x2 UML class as documented in the TransformationMatrix2x2 section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TransformationMatrix2x2 UML class; including the name, definition, type, and cardinality of those documented in the TransformationMatrix2x2 section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TransformationMatrix2x2 UML class; including the name, definition, type, and cardinality of those documented in the TransformationMatrix2x2 section of the <a href="#">Core Data Dictionary</a> .

### **Class 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»

#### **Relations**

#### **Attributes**

### **11.1.35. Class TransformationMatrix3x4**

<b>Requirement 34</b>	<a href="#">/req/Core/TransformationMatrix3x4</a>
A	Any use of the TransformationMatrix3x4 type in the Implementation Specification SHALL have the same definition as the TransformationMatrix3x4 UML class as documented in the TransformationMatrix3x4 section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TransformationMatrix3x4 UML class as documented in the TransformationMatrix3x4 section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TransformationMatrix3x4 UML class; including the name, definition, type, and cardinality of those documented in the TransformationMatrix3x4 section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TransformationMatrix3x4 UML class; including the name, definition, type, and cardinality of those documented in the TransformationMatrix3x4 section of the <a href="#">Core Data Dictionary</a> .

### **Class 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»

#### **Relations**

#### **Attributes**

### **11.1.36. Class TransformationMatrix4x4**

<b>Requirement 35</b>	<a href="#">/req/Core/TransformationMatrix4x4</a>
-----------------------	---------------------------------------------------

A	Any use of the TransformationMatrix4x4 type in the Implementation Specification SHALL have the same definition as the TransformationMatrix4x4 UML class as documented in the TransformationMatrix4x4 section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TransformationMatrix4x4 UML class as documented in the TransformationMatrix4x4 section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TransformationMatrix4x4 UML class; including the name, definition, type, and cardinality of those documented in the TransformationMatrix4x4 section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TransformationMatrix4x4 UML class; including the name, definition, type, and cardinality of those documented in the TransformationMatrix4x4 section of the <a href="#">Core Data Dictionary</a> .

#### Class 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»

#### Relations

#### Attributes

### 11.1.37. Class AbstractGenericAttribute

Requirement 36	/req/Core/AbstractGenericAttribute
A	Any use of the AbstractGenericAttribute type in the Implementation Specification SHALL have the same definition as the AbstractGenericAttribute UML class as documented in the AbstractGenericAttribute section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the AbstractGenericAttribute UML class as documented in the AbstractGenericAttribute section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the AbstractGenericAttribute UML class; including the name, definition, type, and cardinality of those documented in the AbstractGenericAttribute section of the <a href="#">Core Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the AbstractGenericAttribute UML class; including the name, definition, type, and cardinality of those documented in the AbstractGenericAttribute section of the <a href="#">Core Data Dictionary</a> .
---	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class AbstractGenericAttribute

Definition: AbstractGenericAttribute is the abstract superclass for all types of generic attributes.

Subclass Of: <-- section,>>

Stereotype: «DataType»

#### Relations

#### Attributes

### 11.1.38. Class CityModelMember

Requirement 37	/req/Core/CityModelMember
A	Any use of the CityModelMember type in an Implementation Specification SHALL be restricted to the valid values specified in the CityModelMember UML class as documented in the <a href="#">Core Data Dictionary</a> .

### Class 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: <-- section,>>

Stereotype: «Union»

#### Relations

#### Attributes

Attribute Name: appearanceMember

Value Type: AbstractAppearance

Definition: Specifies the appearances of the CityModel.

Multiplicity:

Stereotype: «Property»

Attribute Name: cityObjectMember

Value Type: AbstractCityObject

Definition: Specifies the city objects that are part of the CityModel.

Multiplicity:

Stereotype: «Property»

Attribute Name:	featureMember
Value Type:	AnyFeature
Definition:	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.
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	versionMember
Value Type:	AbstractVersion
Definition:	Specifies the different versions of the CityModel.
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	versionTransitionMember
Value Type:	AbstractVersionTransition
Definition:	Specifies the transitions between the different versions of the CityModel.
Multiplicity:	
Stereotype:	«Property»

### 11.1.39. Class ExternalReference

Requirement 38	/req/Core/ExternalReference
A	Any use of the ExternalReference type in the Implementation Specification SHALL have the same definition as the ExternalReference UML class as documented in the ExternalReference section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ExternalReference UML class as documented in the ExternalReference section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the ExternalReference UML class; including the name, definition, type, and cardinality of those documented in the ExternalReference section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ExternalReference UML class; including the name, definition, type, and cardinality of those documented in the ExternalReference section of the <a href="#">Core Data Dictionary</a> .

## **Class 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: <-- section,>>

Stereotype: «DataType»

## **Relations**

## **Attributes**

Attribute Name: informationSystem

Value Type: URI

Definition: Specifies the URI that points to the external information system.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: relationType

Value Type: URI

Definition: 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.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: targetResource

Value Type: URI

Definition: Specifies the URI that points to the object in the external information system.

Multiplicity:

«Property»

## **11.1.40. Class Occupancy**

Requirement 39	/req/Core/Occupancy
A	Any use of the Occupancy type in the Implementation Specification SHALL have the same definition as the Occupancy UML class as documented in the Occupancy section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Occupancy UML class as documented in the Occupancy section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Occupancy UML class; including the name, definition, type, and cardinality of those documented in the Occupancy section of the <a href="#">Core Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the Occupancy UML class; including the name, definition, type, and cardinality of those documented in the Occupancy section of the <a href="#">Core Data Dictionary</a> .
---	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class Occupancy

Definition: Occupancy is an application-dependent indication of what is contained by a feature.

Subclass Of: <-- section,>>

Stereotype: «DataType»

### Relations

### Attributes

Attribute Name: interval

Value Type: IntervalValue

Definition: Indicates the time period the occupants are contained by a feature.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: numberOfOccupants

Value Type: Integer

Definition: Indicates the number of occupants contained by a feature.

Multiplicity:

Stereotype: «Property»

Attribute Name: occupantType

Value Type: OccupantTypeValue

Definition: Indicates the specific type of the occupants that are contained by a feature.

Multiplicity: [0..1]

Stereotype: «Property»

## 11.1.41. Class QualifiedArea

Requirement 40	/req/Core/QualifiedArea
A	Any use of the QualifiedArea type in the Implementation Specification SHALL have the same definition as the QualifiedArea UML class as documented in the QualifiedArea section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the QualifiedArea UML class as documented in the QualifiedArea section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the QualifiedArea UML class; including the name, definition, type, and cardinality of those documented in the QualifiedArea section of the <a href="#">Core Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the QualifiedArea UML class; including the name, definition, type, and cardinality of those documented in the QualifiedArea section of the <a href="#">Core Data Dictionary</a> .
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class QualifiedArea

Definition: QualifiedArea is an application-dependent measure of the area of a space or of a thematic surface.

Subclass Of: <-- section,>>

Stereotype: «DataType»

#### Relations

#### Attributes

Attribute Name: area

Value Type: Area

Definition: Specifies the value of the QualifiedArea.

Multiplicity:

Stereotype: «Property»

Attribute Name: typeOfArea

Value Type: QualifiedAreaTypeValue

Definition: Indicates the specific type of the QualifiedArea.

Multiplicity:

Stereotype: «Property»

### 11.1.42. Class QualifiedVolume

Requirement 41	/req/Core/QualifiedVolume
A	Any use of the QualifiedVolume type in the Implementation Specification SHALL have the same definition as the QualifiedVolume UML class as documented in the QualifiedVolume section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the QualifiedVolume UML class as documented in the QualifiedVolume section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the QualifiedVolume UML class; including the name, definition, type, and cardinality of those documented in the QualifiedVolume section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the QualifiedVolume UML class; including the name, definition, type, and cardinality of those documented in the QualifiedVolume section of the <a href="#">Core Data Dictionary</a> .

## **Class QualifiedVolume**

Definition: QualifiedVolume is an application-dependent measure of the volume of a space.

Subclass Of: <-- section,>>

Stereotype: «DataType»

## **Relations**

## **Attributes**

Attribute Name: typeOfVolume

Value Type: QualifiedVolumeTypeValue

Definition: Indicates the specific type of the QualifiedVolume.

Multiplicity:

Stereotype: «Property»

Attribute Name: volume

Value Type: Volume

Definition: Specifies the value of the QualifiedVolume.

Multiplicity:

Stereotype: «Property»

## **11.1.43. Class RelativeToTerrain**

Requirement 42	/req/Core/RelativeToTerrain
A	Any use of the RelativeToTerrain type in an Implementation Specification SHALL be restricted to the valid values specified in the RelativeToTerrain UML class as documented in the <a href="#">Core Data Dictionary</a> .

## **Class RelativeToTerrain**

Definition: RelativeToTerrain enumerates the spatial relations of a city object relative to terrain in a qualitative way.

Subclass Of: <-- section,>>

Stereotype:

## **Relations**

## **Attributes**

Attribute Name: entirelyAboveTerrain

Value Type:

Definition: Indicates that the city object is located entirely above the terrain.

Multiplicity:

Stereotype:

Attribute Name: substantiallyAboveTerrain

Value Type:

Definition: Indicates that the city object is for the most part located above the terrain.

Multiplicity:

Stereotype:

Attribute Name:	substantiallyAboveAndBelowTerrain
Value Type:	
Definition:	Indicates that the city object is located half above the terrain and half below the terrain.
Multiplicity:	
Stereotype:	
Attribute Name:	substantiallyBelowTerrain
Value Type:	
Definition:	Indicates that the city object is for the most part located below the terrain.
Multiplicity:	
Stereotype:	
Attribute Name:	entirelyBelowTerrain
Value Type:	
Definition:	Indicates that the city object is located entirely below the terrain.
Multiplicity:	
Stereotype:	

#### 11.1.44. Class RelativeToWater

Requirement 43	/req/Core/RelativeToWater
A	Any use of the RelativeToWater type in an Implementation Specification SHALL be restricted to the valid values specified in the RelativeToWater UML class as documented in the <a href="#">Core Data Dictionary</a> .

##### Class RelativeToWater

Definition:	RelativeToWater enumerates the spatial relations of a city object relative to the water surface in a qualitative way.
Subclass Of:	<-- section,>>
Stereotype:	

##### Relations

##### Attributes

Attribute Name:	entirelyAboveWaterSurface
Value Type:	
Definition:	Indicates that the city object is located entirely above the water surface.
Multiplicity:	
Stereotype:	

Attribute Name:	substantiallyAboveWaterSurface
Value Type:	
Definition:	Indicates that the city object is for the most part located above the water surface.
Multiplicity:	
Stereotype:	

Attribute Name:	substantiallyAboveAndBelowWaterSurface
Value Type:	
Definition:	Indicates that the city object is located half above the water surface and half below the water surface.
Multiplicity:	
Stereotype:	
Attribute Name:	substantiallyBelowWaterSurface
Value Type:	
Definition:	Indicates that the city object is for the most part located below the water surface.
Multiplicity:	
Stereotype:	
Attribute Name:	entirelyBelowWaterSurface
Value Type:	
Definition:	Indicates that the city object is located entirely below the water surface.
Multiplicity:	
Stereotype:	
Attribute Name:	temporarilyAboveAndBelowWaterSurface
Value Type:	
Definition:	Indicates that city object is temporarily located above or below the water level, because the height of the water surface is varying.
Multiplicity:	
Stereotype:	

### 11.1.45. Class SpaceType

Requirement 44	/req/Core/SpaceType
A	Any use of the SpaceType type in an Implementation Specification SHALL be restricted to the valid values specified in the SpaceType UML class as documented in the SpaceType section of the <a href="#">Core Data Dictionary</a> .

#### Class SpaceType

Definition: SpaceType is an enumeration that characterises a space according to its closure properties.

Subclass Of: <-- section,>>

Stereotype:

#### Relations

#### Attributes

Attribute Name:	closed
Value Type:	
Definition:	Indicates that the space has boundaries at the bottom, at the top, and on all sides.
Multiplicity:	
Stereotype:	
Attribute Name:	open
Value Type:	
Definition:	Indicates that the space has at maximum a boundary at the bottom.
Multiplicity:	
Stereotype:	
Attribute Name:	semiOpen
Value Type:	
Definition:	Indicates that the space has a boundary at the bottom and on at least one side.
Multiplicity:	
Stereotype:	

### 11.1.46. Class XALAddressDetails

Requirement 45	/req/Core/XALAddressDetails
A	Any use of the XALAddressDetails type in the Implementation Specification SHALL have the same definition as the XALAddressDetails UML class as documented in the XALAddressDetails section of the <a href="#">Core Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the XALAddressDetails UML class as documented in the XALAddressDetails section of the <a href="#">Core Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the XALAddressDetails UML class; including the name, definition, type, and cardinality of those documented in the XALAddressDetails section of the <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the XALAddressDetails UML class; including the name, definition, type, and cardinality of those documented in the XALAddressDetails section of the <a href="#">Core Data Dictionary</a> .

#### Class XALAddressDetails

Definition:	XALAddressDetails represents address details according to the OASIS xAL standard.
Subclass Of:	<-- section,>>
Stereotype:	«DataType»

#### Relations

## Attributes

### 11.1.47. Additional Information

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

## 11.2. Appearance

### Requirements Class

<http://www.opengis.net/spec/CityGML/3.0/req/req-class-appearance>

Target type	Conceptual Model
Dependency	<a href="/req/req-class-core">/req/req-class-core</a>

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](#).

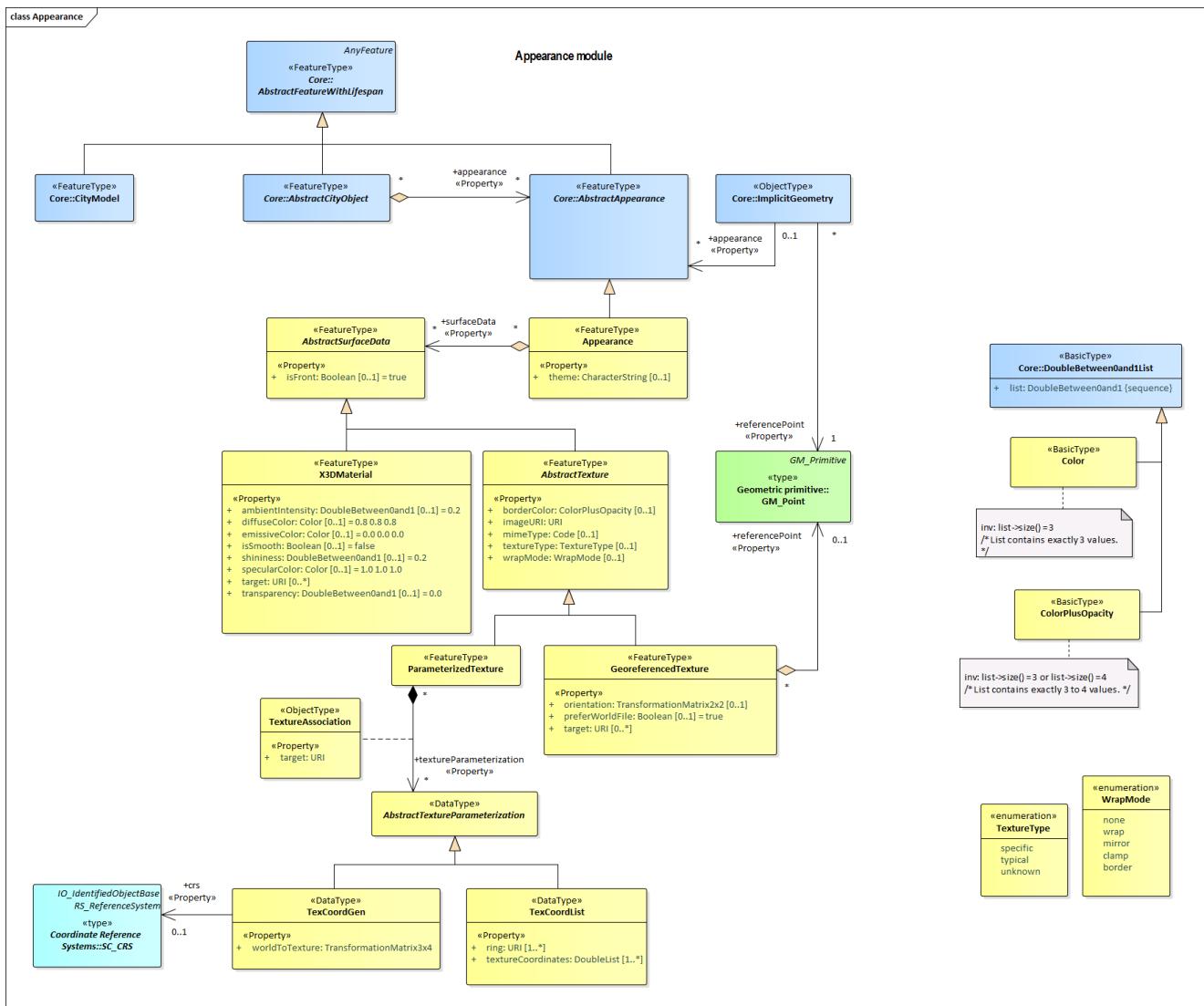


Figure 15. UML diagram of CityGML's appearance model.

### 11.2.1. Appearance 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.

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.2.2. Class AbstractSurfaceData

Requirement 46	/req/Appearance/AbstractSurfaceData
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractSurfaceData UML class as documented in the AbstractSurfaceData section of the <a href="#">Appearance Data Dictionary</a> .

B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractSurfaceData UML class as documented in the AbstractSurfaceData section of the <a href="#">Appearance Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractSurfaceData UML class; including the name, definition, type, and cardinality of those documented in the AbstractSurfaceData section of the <a href="#">Appearance Data Dictionary</a> .

### Class AbstractSurfaceData

Definition: AbstractSurfaceData is the abstract superclass for different kinds of textures and material.

Subclass Of: <-- section,>>

Stereotype: «FeatureType»

### Relations

### Attributes

Attribute Name: isFront

Value Type: Boolean

Definition: Indicates whether the texture or material is assigned to the front side or the back side of the surface geometry object.

Multiplicity: [0..1]

Stereotype: «Property»

### 11.2.3. Class AbstractTexture

Requirement 47	/req/Appearance/AbstractTexture
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractTexture UML class as documented in the AbstractTexture section of the <a href="#">Appearance Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractTexture UML class as documented in the AbstractTexture section of the <a href="#">Appearance Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractTexture UML class; including the name, definition, type, and cardinality of those documented in the AbstractTexture section of the <a href="#">Appearance Data Dictionary</a> .

## **Class AbstractTexture**

Definition: AbstractTexture is the abstract superclass to represent the common attributes of the classes ParameterizedTexture and GeoreferencedTexture.

Subclass Of: [AbstractSurfaceData](#)

Stereotype: «FeatureType»

### **Relations**

### **Attributes**

Attribute Name: borderColor

Value Type: ColorPlusOpacity

Definition: Specifies the color of that part of the surface that is not covered by the texture.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: imageURI

Value Type: URI

Definition: Specifies the URI that points to the external image data file.

Multiplicity:

Stereotype: «Property»

Attribute Name: mimeType

Value Type: Code

Definition: Specifies the MIME type of the external point cloud file.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: textureType

Value Type: TextureType

Definition: Indicates the specific type of the texture.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: wrapMode

Value Type: WrapMode

Definition: Specifies the behaviour of the texture when the texture is smaller than the surface to which it is applied.

Multiplicity: [0..1]

Stereotype: «Property»

### **11.2.4. Class Appearance**

Requirement 48	/req/Appearance/Appearance
A	The Implementation Specification SHALL contain an element with the same definition as the Appearance UML class as documented in the Appearance section of the <a href="#">Appearance Data Dictionary</a> .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Appearance UML class as documented in the Appearance section of the <a href="#">Appearance Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Appearance UML class; including the name, definition, type, and cardinality of those documented in the Appearance section of the <a href="#">Appearance Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Appearance UML class; including the name, definition, type, and cardinality of those documented in the Appearance section of the <a href="#">Appearance Data Dictionary</a> .

### Class 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»

### Relations

Relation Name: surfaceData

Cardinality: \*

Target Class: [AbstractSurfaceData](#)

Definition: Relates to the surface data that are part of the Appearance.

### Attributes

Attribute Name: theme

Value Type: CharacterString

Definition: 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.

Multiplicity: [0..1]

Stereotype: «Property»

## 11.2.5. Class Color

Requirement 49	/req/Appearance/Color
A	Any use of the Color type in the Implementation Specification SHALL have the same definition as the Color UML class as documented in the Color section of the <a href="#">Appearance Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Color UML class as documented in the Color section of the <a href="#">Appearance Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the Color UML class; including the name, definition, type, and cardinality of those documented in the Color section of the <a href="#">Appearance Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Color UML class; including the name, definition, type, and cardinality of those documented in the Color section of the <a href="#">Appearance Data Dictionary</a> .

### Class Color

Definition: Color is a list of three double values between 0 and 1 defining an RGB color value.

Subclass Of: [DoubleBetween0and1List](#)

Stereotype: «BasicType»

### Relations

### Attributes

## 11.2.6. Class ColorPlusOpacity

Requirement 50	/req/Appearance/ColorPlusOpacity
A	Any use of the ColorPlusOpacity type in the Implementation Specification SHALL have the same definition as the ColorPlusOpacity UML class as documented in the ColorPlusOpacity section of the <a href="#">Appearance Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ColorPlusOpacity UML class as documented in the ColorPlusOpacity section of the <a href="#">Appearance Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the ColorPlusOpacity UML class; including the name, definition, type, and cardinality of those documented in the ColorPlusOpacity section of the <a href="#">Appearance Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ColorPlusOpacity UML class; including the name, definition, type, and cardinality of those documented in the ColorPlusOpacity section of the <a href="#">Appearance Data Dictionary</a> .

### Class 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»

### Relations

## Attributes

### 11.2.7. Class GeoreferencedTexture

Requirement 51	/req/Appearance/GeoreferencedTexture
A	The Implementation Specification SHALL contain an element with the same definition as the GeoreferencedTexture UML class as documented in the GeoreferencedTexture section of the <a href="#">Appearance Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GeoreferencedTexture UML class as documented in the GeoreferencedTexture section of the <a href="#">Appearance Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the GeoreferencedTexture UML class; including the name, definition, type, and cardinality of those documented in the GeoreferencedTexture section of the <a href="#">Appearance Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GeoreferencedTexture UML class; including the name, definition, type, and cardinality of those documented in the GeoreferencedTexture section of the <a href="#">Appearance Data Dictionary</a> .

## Class 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 accompanying world file or specified using the orientation and referencePoint elements.

Subclass Of: [AbstractTexture](#)

Stereotype: «FeatureType»

## Relations

Relation Name: referencePoint

Cardinality: 0..1

Target Class: [GM\\_Point](#)

Definition: Relates to the 2D Point Geometry that represents the center of the upper left image pixel in world space.

## Attributes

Attribute Name: orientation

Value Type: TransformationMatrix2x2

Definition: Specifies the rotation and scaling of the image in form of a 2x2 matrix.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	preferWorldFile
Value Type:	Boolean
Definition:	Indicates whether the georeference from the image file or the accompanying world file should be preferred.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	target
Value Type:	URI
Definition:	Specifies the URI that points to the surface geometry objects to which the texture is applied.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.2.8. Class ParameterizedTexture

Requirement 52	/req/Appearance/ParameterizedTexture
A	The Implementation Specification SHALL contain an element with the same definition as the ParameterizedTexture UML class as documented in the ParameterizedTexture section of the <a href="#">Appearance Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ParameterizedTexture UML class as documented in the ParameterizedTexture section of the <a href="#">Appearance Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the ParameterizedTexture UML class; including the name, definition, type, and cardinality of those documented in the ParameterizedTexture section of the <a href="#">Appearance Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ParameterizedTexture UML class; including the name, definition, type, and cardinality of those documented in the ParameterizedTexture section of the <a href="#">Appearance Data Dictionary</a> .

#### Class ParameterizedTexture

Definition: A ParameterizedTexture is a texture that uses texture coordinates or a transformation matrix for parameterization.

Subclass Of: [AbstractTexture](#)

Stereotype: «FeatureType»

#### Relations

Relation Name: textureParameterization  
 Cardinality: \*  
 Target Class: [AbstractTextureParameterization](#)  
 Definition: Relates to the texture coordinates or transformation matrices used for parameterization.

#### Attributes

### 11.2.9. Class TextureAssociation

Requirement 53	/req/Appearance/TextureAssociation
A	The Implementation Specification SHALL contain an element with the same definition as the TextureAssociation UML class as documented in the TextureAssociation section of the <a href="#">Appearance Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TextureAssociation UML class as documented in the TextureAssociation section of the <a href="#">Appearance Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TextureAssociation UML class; including the name, definition, type, and cardinality of those documented in the TextureAssociation section of the <a href="#">Appearance Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TextureAssociation UML class; including the name, definition, type, and cardinality of those documented in the TextureAssociation section of the <a href="#">Appearance Data Dictionary</a> .

#### Class TextureAssociation

Definition: TextureAssociation denotes the relation of a texture to a surface geometry object.  
 Subclass Of: <-- section,>>  
 Stereotype: «ObjectType»

#### Relations

#### Attributes

Attribute Name: target  
 Value Type: URI  
 Definition: Specifies the URI that points to the surface geometry object to which the texture is applied.  
 Multiplicity:  
 Stereotype: «Property»

## 11.2.10. Class X3DMaterial

Requirement 54	/req/Appearance/X3DMaterial
A	The Implementation Specification SHALL contain an element with the same definition as the X3DMaterial UML class as documented in the X3DMaterial section of the <a href="#">Appearance Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the X3DMaterial UML class as documented in the X3DMaterial section of the <a href="#">Appearance Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the X3DMaterial UML class; including the name, definition, type, and cardinality of those documented in the X3DMaterial section of the <a href="#">Appearance Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the X3DMaterial UML class; including the name, definition, type, and cardinality of those documented in the X3DMaterial section of the <a href="#">Appearance Data Dictionary</a> .

### Class 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»

### Relations

### Attributes

Attribute Name:	ambientIntensity
Value Type:	DoubleBetween0and1
Definition:	Specifies the minimum percentage of diffuseColor that is visible regardless of light sources.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	diffuseColor
Value Type:	Color
Definition:	Specifies the color of the light diffusely reflected by the surface geometry object.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	emissiveColor
Value Type:	Color
Definition:	Specifies the color of the light emitted by the surface geometry object.
Multiplicity:	[0..1]
Stereotype:	«Property»

Attribute Name:	isSmooth
Value Type:	Boolean
Definition:	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).
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	shininess
Value Type:	DoubleBetween0and1
Definition:	Specifies the sharpness of the specular highlight.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	specularColor
Value Type:	Color
Definition:	Specifies the color of the light directly reflected by the surface geometry object.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	target
Value Type:	URI
Definition:	Specifies the URI that points to the surface geometry objects to which the material is applied.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	transparency
Value Type:	DoubleBetween0and1
Definition:	Specifies the degree of transparency of the surface geometry object.
Multiplicity:	[0..1]
Stereotype:	«Property»

### 11.2.11. Class AbstractTextureParameterization

Requirement 55	/req/Appearance/AbstractTextureParameterization
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractTextureParameterization UML class as documented in the AbstractTextureParameterization section of the <a href="#">Appearance Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractTextureParameterization UML class as documented in the AbstractTextureParameterization section of the <a href="#">Appearance Data Dictionary</a> .

C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractTextureParameterization UML class; including the name, definition, type, and cardinality of those documented in the AbstractTextureParameterization section of the <a href="#">Appearance Data Dictionary</a> .
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class AbstractTextureParameterization

Definition: AbstractTextureParameterization is the abstract superclass for different kinds of texture parameterizations.

Subclass Of: <-->

Stereotype: «DataType»

#### Relations

#### Attributes

### 11.2.12. Class TexCoordGen

Requirement 56	/req/Appearance/TexCoordGen
A	Any use of the TexCoordGen type in the Implementation Specification SHALL have the same definition as the TexCoordGen UML class as documented in the TexCoordGen section of the <a href="#">Appearance Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TexCoordGen UML class as documented in the TexCoordGen section of the <a href="#">Appearance Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TexCoordGen UML class; including the name, definition, type, and cardinality of those documented in the TexCoordGen section of the <a href="#">Appearance Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TexCoordGen UML class; including the name, definition, type, and cardinality of those documented in the TexCoordGen section of the <a href="#">Appearance Data Dictionary</a> .

### Class TexCoordGen

Definition: TexCoordGen defines texture parameterization using a transformation matrix.

Subclass Of: <-->

Stereotype: «DataType»

#### Relations

Relation Name:	crs
Cardinality:	0..1
Target Class:	<a href="#">SC_CRS</a>
Definition:	Relates to the coordinate reference system of the transformation matrix.
<b>Attributes</b>	
Attribute Name:	worldToTexture
Value Type:	TransformationMatrix3x4
Definition:	Specifies the 3x4 transformation matrix that defines the transformation between world coordinates and texture coordinates.
Multiplicity:	
Stereotype:	«Property»

### 11.2.13. Class TexCoordList

Requirement 57	/req/Appearance/TexCoordList
A	Any use of the TexCoordList type in the Implementation Specification SHALL have the same definition as the TexCoordList UML class as documented in the TexCoordList section of the <a href="#">Appearance Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TexCoordList UML class as documented in the TexCoordList section of the <a href="#">Appearance Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TexCoordList UML class; including the name, definition, type, and cardinality of those documented in the TexCoordList section of the <a href="#">Appearance Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TexCoordList UML class; including the name, definition, type, and cardinality of those documented in the TexCoordList section of the <a href="#">Appearance Data Dictionary</a> .

#### Class TexCoordList

Definition: TexCoordList defines texture parameterization using texture coordinates.  
 Subclass Of: <- section,>>  
 Stereotype: «DataType»

#### Relations

#### Attributes

Attribute Name:	ring
Value Type:	URI
Definition:	Specifies the URIs that point to the LinearRings that are parameterized using the given texture coordinates.
Multiplicity:	[1..*]
Stereotype:	«Property»
Attribute Name:	textureCoordinates
Value Type:	DoubleList
Definition:	Specifies the coordinates of texture used for parameterization. The texture coordinates are provided separately for each LinearRing of the surface geometry object.
Multiplicity:	[1..*]
Stereotype:	«Property»

## 11.2.14. Class TextureType

Requirement 58	/req/Appearance/TextureType
A	Any use of the TextureType type in the Implementation Specification SHALL have the same definition as the TextureType UML class as documented in the TextureType section of the <a href="#">Appearance Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TextureType UML class as documented in the TextureType section of the <a href="#">Appearance Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TextureType UML class; including the name, definition, type, and cardinality of those documented in the TextureType section of the <a href="#">Appearance Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TextureType UML class; including the name, definition, type, and cardinality of those documented in the TextureType section of the <a href="#">Appearance Data Dictionary</a> .

### Class TextureType

Definition: TextureType enumerates the different texture types.

Subclass Of: <-> section,>>

Stereotype:

### Relations

### Attributes

Attribute Name: specific

Value Type:

Definition: Indicates that the texture is specific to a single surface.

Multiplicity:

Stereotype:

Attribute Name: typical

Value Type:

Definition: Indicates that the texture is characteristic of a surface and can be used repeatedly.

Multiplicity:

Stereotype:

Attribute Name: unknown

Value Type:

Definition: Indicates that the texture type is not known.

Multiplicity:

Stereotype:

### 11.2.15. Class WrapMode

Requirement 59	/req/Appearance/WrapMode
A	Any use of the WrapMode type in the Implementation Specification SHALL have the same definition as the WrapMode UML class as documented in the WrapMode section of the <a href="#">Appearance Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the WrapMode UML class as documented in the WrapMode section of the <a href="#">Appearance Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the WrapMode UML class; including the name, definition, type, and cardinality of those documented in the WrapMode section of the <a href="#">Appearance Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the WrapMode UML class; including the name, definition, type, and cardinality of those documented in the WrapMode section of the <a href="#">Appearance Data Dictionary</a> .

#### Class WrapMode

Definition: WrapMode enumerates the different fill modes for textures.

Subclass Of: <--> section,>>

Stereotype:

#### Relations

#### Attributes

Attribute Name:	none
Value Type:	
Definition:	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]
Multiplicity:	
Stereotype:	
Attribute Name:	wrap
Value Type:	
Definition:	Indicates that the texture is repeated until the surface is fully covered. [cf. COLLADA]
Multiplicity:	
Stereotype:	
Attribute Name:	mirror
Value Type:	
Definition:	Indicates that the texture is repeated and mirrored. [cf. COLLADA]
Multiplicity:	
Stereotype:	
Attribute Name:	clamp
Value Type:	
Definition:	Indicates that the texture is stretched to the edges of the surface. [cf. COLLADA]
Multiplicity:	
Stereotype:	
Attribute Name:	border
Value Type:	
Definition:	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]
Multiplicity:	
Stereotype:	

## 11.2.16. Additional Information

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

## 11.3. Bridge

### Requirements Class

<http://www.opengis.net/spec/CityGML/3.0/req/req-class-bridge>

Target type	Conceptual Model
Dependency	<a href="/req/req-class-core">/req/req-class-core</a>

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](#).

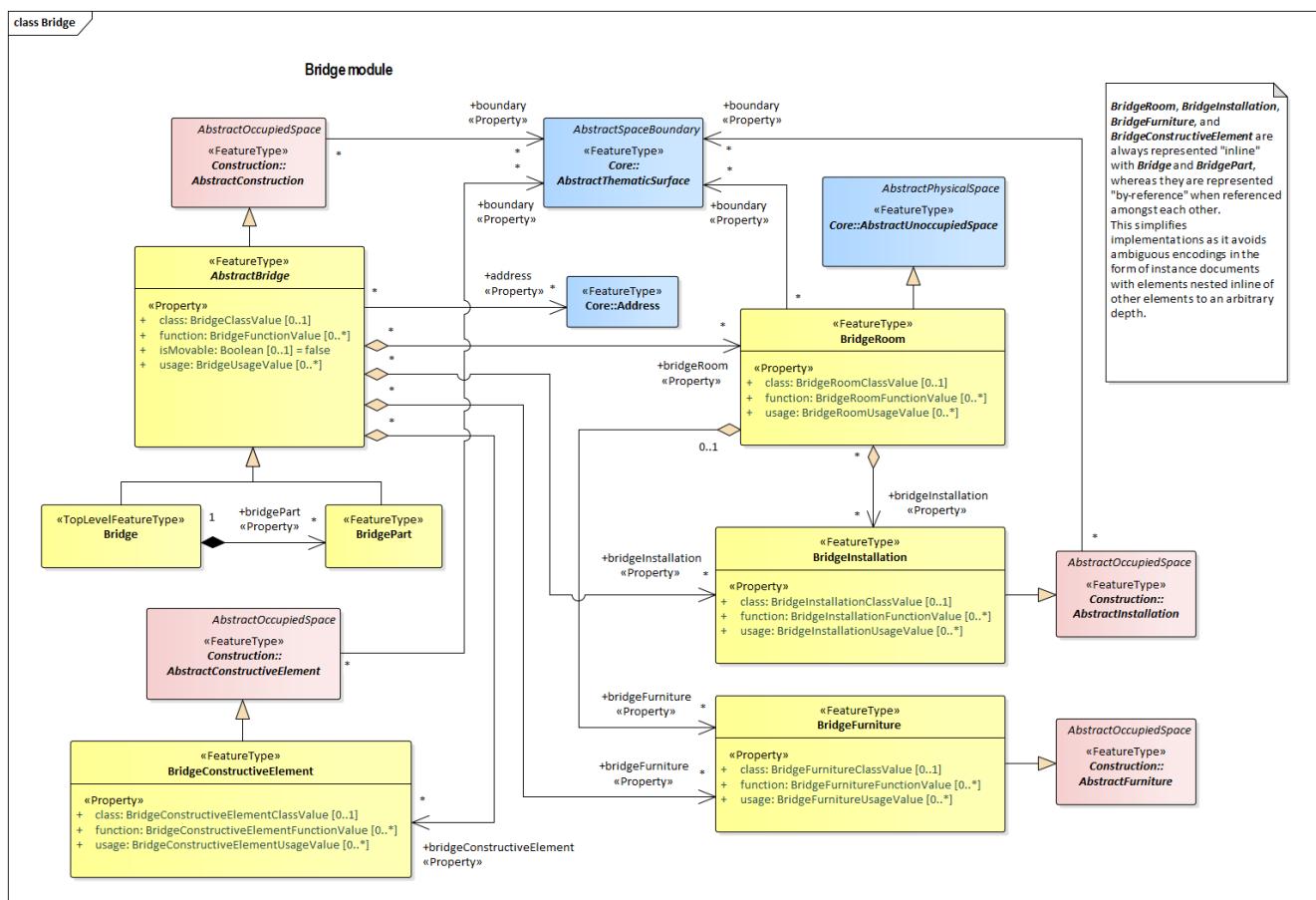


Figure 16. UML diagram of the Bridge Model.

### 11.3.1. Bridge Package

#### Package Bridge

Description: The Bridge module supports representation of thematic and spatial aspects of bridges, bridge parts, bridge installations, and interior bridge structures.

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.3.2. Class AbstractBridge

Requirement 60	/req/Bridge/AbstractBridge
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractBridge UML class as documented in the AbstractBridge section of the <a href="#">Bridge Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractBridge UML class as documented in the AbstractBridge section of the <a href="#">Bridge Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractBridge UML class; including the name, definition, type, and cardinality of those documented in the AbstractBridge section of the <a href="#">Bridge Data Dictionary</a> .

#### Class AbstractBridge

Definition: AbstractBridge is an abstract superclass representing the common attributes and associations of the classes Bridge and BridgePart.

Subclass Of: [AbstractConstruction](#)

Stereotype: «FeatureType»

#### Relations

Relation Name: bridgeRoom

Cardinality: \*

Target Class: [BridgeRoom](#)

Definition: Relates the rooms to the Bridge or BridgePart.

Relation Name: address

Cardinality: \*

Target Class: [Address](#)

Definition: Relates the addresses to the Bridge or BridgePart.

Relation Name: bridgeConstructiveElement

Cardinality: \*

Target Class: [BridgeConstructiveElement](#)

Definition: Relates the constructive elements to the Bridge or BridgePart.

Relation Name:	bridgeInstallation
Cardinality:	*
Target Class:	<a href="#">BridgeInstallation</a>
Definition:	Relates the installation objects to the Bridge or BridgePart.
Relation Name:	bridgeFurniture
Cardinality:	*
Target Class:	<a href="#">BridgeFurniture</a>
Definition:	Relates the furniture objects to the Bridge or BridgePart.
<b>Attributes</b>	
Attribute Name:	class
Value Type:	BridgeClassValue
Definition:	Indicates the specific type of the Bridge or BridgePart.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	BridgeFunctionValue
Definition:	Specifies the intended purposes of the Bridge or BridgePart.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	isMovable
Value Type:	Boolean
Definition:	Indicates whether the Bridge or BridgePart can be moved to allow for watercraft to pass.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BridgeUsageValue
Definition:	Specifies the actual uses of the Bridge or BridgePart.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.3.3. Class Bridge

Requirement 61	/req/Bridge/Bridge
A	The Implementation Specification SHALL contain an element with the same definition as the Bridge UML class as documented in the Bridge section of the <a href="#">Bridge Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Bridge UML class as documented in the Bridge section of the <a href="#">Bridge Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the Bridge UML class; including the name, definition, type, and cardinality of those documented in the Bridge section of the <a href="#">Bridge Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Bridge UML class; including the name, definition, type, and cardinality of those documented in the Bridge section of the <a href="#">Bridge Data Dictionary</a> .

### Class 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»

### Relations

Relation Name: bridgePart

Cardinality: \*

Target Class: [BridgePart](#)

Definition: Relates to the bridge parts.

### Attributes

## 11.3.4. Class BridgeClassValue

Requirement 62	/req/Bridge/BridgeClassValue
A	Any use of the BridgeClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeClassValue UML class as documented in the BridgeClassValue section of the <a href="#">Bridge Data Dictionary</a> .

### Class BridgeClassValue

Definition: BridgeClassValue is a code list used to further classify a Bridge.

Subclass Of: <-> section,>>

Stereotype: «CodeList»

### Relations

### Attributes

## 11.3.5. Class BridgeConstructiveElement

Requirement 63	/req/Bridge/BridgeConstructiveElement
----------------	---------------------------------------

A	The Implementation Specification SHALL contain an element with the same definition as the BridgeConstructiveElement UML class as documented in the BridgeConstructiveElement section of the <a href="#">Bridge Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BridgeConstructiveElement UML class as documented in the BridgeConstructiveElement section of the <a href="#">Bridge Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the BridgeConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the BridgeConstructiveElement section of the <a href="#">Bridge Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BridgeConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the BridgeConstructiveElement section of the <a href="#">Bridge Data Dictionary</a> .

### Class 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»

### Relations

### Attributes

Attribute Name: class

Value Type: BridgeConstructiveElementClassName

Definition: Indicates the specific type of the BridgeConstructiveElement.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: BridgeConstructiveElementFunctionName

Definition: Specifies the intended purposes of the BridgeConstructiveElement.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: BridgeConstructiveElementUsageName

Definition: Specifies the actual uses of the BridgeConstructiveElement.

Multiplicity: [0..\*]

Stereotype: «Property»

### 11.3.6. Class BridgeConstructiveElementClassValue

Requirement 64	/req/Bridge/BridgeConstructiveElementClassValue
A	Any use of the BridgeConstructiveElementClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeConstructiveElementClassValue UML class as documented in the BridgeConstructiveElementClassValue section of the <a href="#">Bridge Data Dictionary</a> .

#### Class BridgeConstructiveElementClassValue

Definition: BridgeConstructiveElementClassValue is a code list used to further classify a BridgeConstructiveElement.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.3.7. Class BridgeConstructiveElementFunctionValue

Requirement 65	/req/Bridge/BridgeConstructiveElementFunctionValue
A	Any use of the BridgeConstructiveElementFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeConstructiveElementFunctionValue UML class as documented in the BridgeConstructiveElementFunctionValue section of the <a href="#">Bridge Data Dictionary</a> .

#### Class BridgeConstructiveElementFunctionValue

Definition: BridgeConstructiveElementFunctionValue is a code list that enumerates the different purposes of a BridgeConstructiveElement.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.3.8. Class BridgeConstructiveElementUsageValue

Requirement 66	/req/Bridge/BridgeConstructiveElementUsageValue
----------------	-------------------------------------------------

A	Any use of the BridgeConstructiveElementUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeConstructiveElementUsageValue UML class as documented in the BridgeConstructiveElementUsageValue section of the <a href="#">Bridge Data Dictionary</a> .
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class BridgeConstructiveElementUsageValue

Definition: BridgeConstructiveElementUsageValue is a code list that enumerates the different uses of a BridgeConstructiveElement.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.3.9. Class BridgeFunctionValue

Requirement 67	/req/Bridge/BridgeFunctionValue
A	Any use of the BridgeFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeFunctionValue UML class as documented in the BridgeFunctionValue section of the <a href="#">Bridge Data Dictionary</a> .

### Class BridgeFunctionValue

Definition: BridgeFunctionValue is a code list that enumerates the different purposes of a Bridge.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.3.10. Class BridgeFurniture

Requirement 68	/req/Bridge/BridgeFurniture
A	The Implementation Specification SHALL contain an element with the same definition as the BridgeFurniture UML class as documented in the BridgeFurniture section of the <a href="#">Bridge Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BridgeFurniture UML class as documented in the BridgeFurniture section of the <a href="#">Bridge Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the BridgeFurniture UML class; including the name, definition, type, and cardinality of those documented in the BridgeFurniture section of the <a href="#">Bridge Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BridgeFurniture UML class; including the name, definition, type, and cardinality of those documented in the BridgeFurniture section of the <a href="#">Bridge Data Dictionary</a> .

### Class 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»

### Relations

### Attributes

Attribute Name: class

Value Type: BridgeFurnitureClassValue

Definition: Indicates the specific type of the BridgeFurniture.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: BridgeFurnitureFunctionValue

Definition: Specifies the intended purposes of the BridgeFurniture.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: BridgeFurnitureUsageValue

Definition: Specifies the actual uses of the BridgeFurniture.

Multiplicity: [0..\*]

Stereotype: «Property»

### 11.3.11. Class BridgeFurnitureClassValue

Requirement 69	/req/Bridge/BridgeFurnitureClassValue
A	Any use of the BridgeFurnitureClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeFurnitureClassValue UML class as documented in the BridgeFurnitureClassValue section of the <a href="#">Bridge Data Dictionary</a> .

### **Class BridgeFurnitureClassValue**

Definition: BridgeFurnitureClassValue is a code list used to further classify a BridgeFurniture.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.3.12. Class BridgeFurnitureFunctionValue**

Requirement 70	/req/Bridge/BridgeFurnitureFunctionValue
A	Any use of the BridgeFurnitureFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeFurnitureFunctionValue UML class as documented in the BridgeFurnitureFunctionValue section of the <a href="#">Bridge Data Dictionary</a> .

### **Class BridgeFurnitureFunctionValue**

Definition: BridgeFurnitureFunctionValue is a code list that enumerates the different purposes of a BridgeFurniture.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.3.13. Class BridgeFurnitureUsageValue**

Requirement 71	/req/Bridge/BridgeFurnitureUsageValue
A	Any use of the BridgeFurnitureUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeFurnitureUsageValue UML class as documented in the BridgeFurnitureUsageValue section of the <a href="#">Bridge Data Dictionary</a> .

### **Class BridgeFurnitureUsageValue**

Definition: BridgeFurnitureUsageValue is a code list that enumerates the different uses of a BridgeFurniture.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### 11.3.14. Class BridgeInstallation

Requirement 72	/req/Bridge/BridgeInstallation
A	The Implementation Specification SHALL contain an element with the same definition as the BridgeInstallation UML class as documented in the BridgeInstallation section of the <a href="#">Bridge Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BridgeInstallation UML class as documented in the BridgeInstallation section of the <a href="#">Bridge Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the BridgeInstallation UML class; including the name, definition, type, and cardinality of those documented in the BridgeInstallation section of the <a href="#">Bridge Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BridgeInstallation UML class; including the name, definition, type, and cardinality of those documented in the BridgeInstallation section of the <a href="#">Bridge Data Dictionary</a> .

#### Class 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»

#### Relations

#### Attributes

Attribute Name: class

Value Type: BridgeInstallationClassValue

Definition: Indicates the specific type of the BridgeInstallation.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: BridgeInstallationFunctionValue

Definition: Specifies the intended purposes of the BridgeInstallation.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name:	usage
Value Type:	BridgeInstallationUsageValue
Definition:	Specifies the actual uses of the BridgeInstallation.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.3.15. Class BridgeInstallationClassValue

Requirement 73	/req/Bridge/BridgeInstallationClassValue
A	Any use of the BridgeInstallationClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeInstallationClassValue UML class as documented in the BridgeInstallationClassValue section of the <a href="#">Bridge Data Dictionary</a> .

#### Class BridgeInstallationClassValue

Definition: BridgeInstallationClassValue is a code list used to further classify a BridgeInstallation.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.3.16. Class BridgeInstallationFunctionValue

Requirement 74	/req/Bridge/BridgeInstallationFundtionValue
A	Any use of the BridgeInstallationFundtionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeInstallationFundtionValue UML class as documented in the BridgeInstallationFundtionValue section of the <a href="#">Bridge Data Dictionary</a> .

#### Class BridgeInstallationFunctionValue

Definition: BridgeInstallationFunctionValue is a code list that enumerates the different purposes of a BridgeInstallation.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.3.17. Class BridgeInstallationUsageValue

<b>Requirement 75</b>	/req/Bridge/BridgeInstallationUsageValue
A	Any use of the BridgeInstallationUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeInstallationUsageValue UML class as documented in the BridgeInstallationUsageValue section of the <a href="#">Bridge Data Dictionary</a> .

### Class BridgeInstallationUsageValue

Definition: BridgeInstallationUsageValue is a code list that enumerates the different uses of a BridgeInstallation.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.3.18. Class BridgePart

<b>Requirement 76</b>	/req/Bridge/BridgePart
A	The Implementation Specification SHALL contain an element with the same definition as the BridgePart UML class as documented in the BridgePart section of the <a href="#">Bridge Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BridgePart UML class as documented in the BridgePart section of the <a href="#">Bridge Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the BridgePart UML class; including the name, definition, type, and cardinality of those documented in the BridgePart section of the <a href="#">Bridge Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BridgePart UML class; including the name, definition, type, and cardinality of those documented in the BridgePart section of the <a href="#">Bridge Data Dictionary</a> .

### Class 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»

#### Relations

#### Attributes

### 11.3.19. Class BridgeRoom

Requirement 77	/req/Bridge/BridgeRoom
A	The Implementation Specification SHALL contain an element with the same definition as the BridgeRoom UML class as documented in the BridgeRoom section of the <a href="#">Bridge Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BridgeRoom UML class as documented in the BridgeRoom section of the <a href="#">Bridge Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the BridgeRoom UML class; including the name, definition, type, and cardinality of those documented in the BridgeRoom section of the <a href="#">Bridge Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BridgeRoom UML class; including the name, definition, type, and cardinality of those documented in the BridgeRoom section of the <a href="#">Bridge Data Dictionary</a> .

#### Class 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»

#### Relations

Relation Name: bridgeInstallation

Cardinality: \*

Target Class: [BridgeInstallation](#)

Definition: Relates to the installation objects.

Relation Name: boundary

Cardinality: \*

Target Class: [AbstractThematicSurface](#)

Definition:

Relation Name: bridgeFurniture

Cardinality: \*

Target Class: [BridgeFurniture](#)

Definition: Relates to the furniture objects.

#### Attributes

Attribute Name:	class
Value Type:	BridgeRoomClassValue
Definition:	Indicates the specific type of the BridgeRoom.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	BridgeRoomFunctionValue
Definition:	Specifies the intended purposes of the BridgeRoom.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BridgeRoomUsageValue
Definition:	Specifies the actual uses of the BridgeRoom.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.3.20. Class BridgeRoomClassValue

Requirement 78	/req/Bridge/BridgeRoomClassValue
A	Any use of the BridgeRoomClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeRoomClassValue UML class as documented in the <a href="#">Bridge Data Dictionary</a> .

#### Class BridgeRoomClassValue

Definition: BridgeRoomClassValue is a code list used to further classify a BridgeRoom.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.3.21. Class BridgeRoomFunctionValue

Requirement 79	/req/Bridge/BridgeRoomFunctionValue
A	Any use of the BridgeRoomFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeRoomFunctionValue UML class as documented in the BridgeRoomFunctionValue section of the <a href="#">Bridge Data Dictionary</a> .

### **Class BridgeRoomFunctionValue**

Definition: BridgeRoomFunctionValue is a code list that enumerates the different purposes of a BridgeRoom.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.3.22. Class BridgeRoomUsageValue**

<b>Requirement 80</b>	<b>/req/Bridge/BridgeRoomUsageValue</b>
A	Any use of the BridgeRoomUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeRoomUsageValue UML class as documented in the BridgeRoomUsageValue section of the <a href="#">Bridge Data Dictionary</a> .

### **Class BridgeRoomUsageValue**

Definition: BridgeRoomUsageValue is a code list that enumerates the different uses of a BridgeRoom.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.3.23. Class BridgeUsageValue**

<b>Requirement 81</b>	<b>/req/Bridge/BridgeUsageValue</b>
A	Any use of the BridgeUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeUsageValue UML class as documented in the BridgeUsageValue section of the <a href="#">Bridge Data Dictionary</a> .

### **Class BridgeUsageValue**

Definition: BridgeUsageValue is a code list that enumerates the different uses of a Bridge.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.3.24. Additional Information**

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

the UML Model.

## 11.4. Building

Requirements Class	
<a href="http://www.opengis.net/spec/CityGML/3.0/req/req-class-building">http://www.opengis.net/spec/CityGML/3.0/req/req-class-building</a>	
Target type	Conceptual Model
Dependency	<a href="#">/req/req-class-core</a>

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](#).

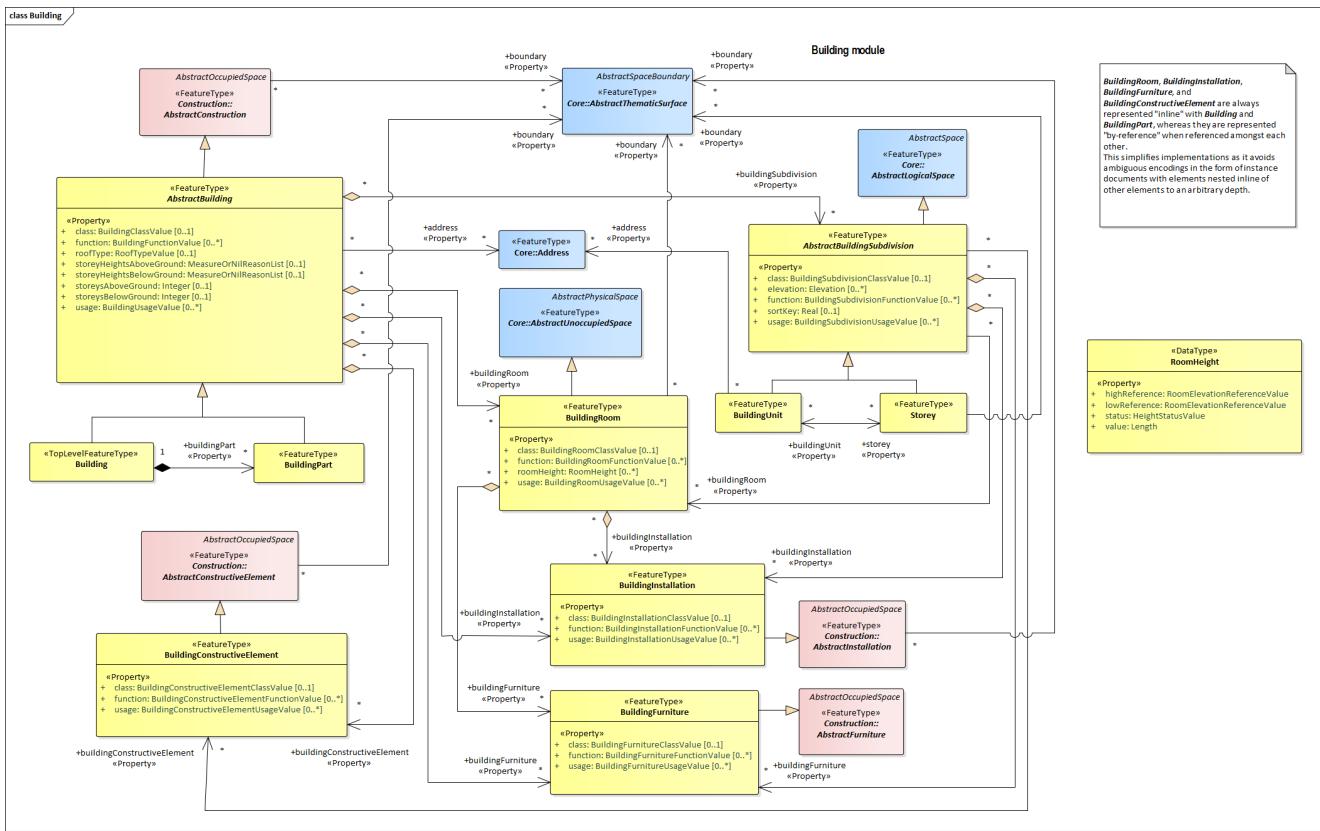


Figure 17. UML diagram of CityGML's building model.

### 11.4.1. Building 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.

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.4.2. Class AbstractBuilding

Requirement 82	/req/Building/AbstractBuilding
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the <b>AbstractBuilding</b> UML class as documented in the <b>AbstractBuilding</b> section of the <a href="#">Building Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the <b>AbstractBuilding</b> UML class as documented in the <b>AbstractBuilding</b> section of the <a href="#">Building Data Dictionary</a> .

C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractBuilding UML class; including the name, definition, type, and cardinality of those documented in the AbstractBuilding section of the <a href="#">Building Data Dictionary</a> .
---	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class AbstractBuilding

Definition: AbstractBuilding is an abstract superclass representing the common attributes and associations of the classes Building and BuildingPart.

Subclass Of: [AbstractConstruction](#)

Stereotype: «FeatureType»

### Relations

Relation Name: buildingConstructiveElement

Cardinality: \*

Target Class: [BuildingConstructiveElement](#)

Definition: Relates the constructive elements to the Building or BuildingPart.

Relation Name: address

Cardinality: \*

Target Class: [Address](#)

Definition: Relates the addresses to the Building or BuildingPart.

Relation Name: buildingSubdivision

Cardinality: \*

Target Class: [AbstractBuildingSubdivision](#)

Definition: Relates the logical subdivisions to the Building or BuildingPart.

Relation Name: buildingFurniture

Cardinality: \*

Target Class: [BuildingFurniture](#)

Definition: Relates the furniture objects to the Building or BuildingPart.

Relation Name: buildingRoom

Cardinality: \*

Target Class: [BuildingRoom](#)

Definition: Relates the rooms to the Building or BuildingPart.

Relation Name: buildingInstallation

Cardinality: \*

Target Class: [BuildingInstallation](#)

Definition: Relates the installation objects to the Building or BuildingPart.

### Attributes

Attribute Name: class

Value Type: [BuildingClassValue](#)

Definition: Indicates the specific type of the Building or BuildingPart.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	function
Value Type:	BuildingFunctionValue
Definition:	Specifies the intended purposes of the Building or BuildingPart.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	roofType
Value Type:	RoofTypeValue
Definition:	Indicates the shape of the roof of the Building or BuildingPart.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	storeyHeightsAboveGround
Value Type:	MeasureOrNilReasonList
Definition:	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.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	storeyHeightsBelowGround
Value Type:	MeasureOrNilReasonList
Definition:	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.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	storeysAboveGround
Value Type:	Integer
Definition:	Indicates the number of storeys positioned above ground level.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	storeysBelowGround
Value Type:	Integer
Definition:	Indicates the number of storeys positioned below ground level.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BuildingUsageValue
Definition:	Specifies the actual uses of the Building or BuildingPart.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.4.3. Class AbstractBuildingSubdivision

Requirement 83	/req/Building/AbstractBuildingSubdivision
----------------	-------------------------------------------

A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractBuildingSubdivision UML class as documented in the AbstractBuildingSubdivision section of the <a href="#">Building Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractBuildingSubdivision UML class as documented in the AbstractBuildingSubdivision section of the <a href="#">Building Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractBuildingSubdivision UML class; including the name, definition, type, and cardinality of those documented in the AbstractBuildingSubdivision section of the <a href="#">Building Data Dictionary</a> .

### Class AbstractBuildingSubdivision

Definition: AbstractBuildingSubdivision is the abstract superclass for different kinds of logical building subdivisions.

Subclass Of: [AbstractLogicalSpace](#)

Stereotype: «FeatureType»

### Relations

Relation Name: buildingInstallation

Cardinality: \*

Target Class: [BuildingInstallation](#)

Definition: Relates to the installation objects.

Relation Name: buildingFurniture

Cardinality: \*

Target Class: [BuildingFurniture](#)

Definition: Relates to the furniture objects.

Relation Name: buildingRoom

Cardinality: \*

Target Class: [BuildingRoom](#)

Definition: Relates to the rooms.

Relation Name: buildingConstructiveElement

Cardinality: \*

Target Class: [BuildingConstructiveElement](#)

Definition: Relates to the constructive elements.

### Attributes

Attribute Name:	class
Value Type:	BuildingSubdivisionClassValue
Definition:	Indicates the specific type of the building subdivision.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	elevation
Value Type:	Elevation
Definition:	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]
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	BuildingSubdivisionFunctionValue
Definition:	Specifies the intended purposes of the building subdivision.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	sortKey
Value Type:	Real
Definition:	Defines an order among the objects that belong to the building subdivision. An example is the sorting of storeys.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BuildingSubdivisionUsageValue
Definition:	Specifies the actual uses of the building subdivision.
Multiplicity:	[0..*]
Stereotype:	«Property»

#### 11.4.4. Class Building

Requirement 84	/req/Building/Building
A	The Implementation Specification SHALL contain an element with the same definition as the Building UML class as documented in the Building section of the <a href="#">Building Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Building UML class as documented in the Building section of the <a href="#">Building Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Building UML class; including the name, definition, type, and cardinality of those documented in the Building section of the <a href="#">Building Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the Building UML class; including the name, definition, type, and cardinality of those documented in the Building section of the <a href="#">Building Data Dictionary</a> .
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class 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»

### Relations

Relation Name: buildingPart

Cardinality: \*

Target Class: [BuildingPart](#)

Definition: Relates to the building parts.

### Attributes

## 11.4.5. Class BuildingClassValue

Requirement 85	/req/Building/BuildingClassValue
A	Any use of the BuildingClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingClassValue UML class as documented in the BuildingClassValue section of the <a href="#">Building Data Dictionary</a> .

### Class BuildingClassValue

Definition: BuildingClassValue is a code list used to further classify a Building.

Subclass Of: <– section,>>

Stereotype: «CodeList»

### Relations

### Attributes

## 11.4.6. Class BuildingConstructiveElement

Requirement 86	/req/Building/BuildingConstructiveElement
A	The Implementation Specification SHALL contain an element with the same definition as the BuildingConstructiveElement UML class as documented in the BuildingConstructiveElement section of the <a href="#">Building Data Dictionary</a> .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BuildingConstructiveElement UML class as documented in the BuildingConstructiveElement section of the <a href="#">Building Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the BuildingConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the BuildingConstructiveElement section of the <a href="#">Building Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BuildingConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the BuildingConstructiveElement section of the <a href="#">Building Data Dictionary</a> .

### Class 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»

### Relations

### Attributes

Attribute Name: class

Value Type: BuildingConstructiveElementClassValue

Definition: Indicates the specific type of the BuildingConstructiveElement.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: BuildingConstructiveElementFunctionValue

Definition: Specifies the intended purposes of the BuildingConstructiveElement.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: BuildingConstructiveElementUsageValue

Definition: Specifies the actual uses of the BuildingConstructiveElement.

Multiplicity: [0..\*]

Stereotype: «Property»

### 11.4.7. Class BuildingConstructiveElementClassValue

Requirement 87	/req/Building/BuildingConstructiveElementClassValue
----------------	-----------------------------------------------------

A	Any use of the BuildingConstructiveElementClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingConstructiveElementClassValue UML class as documented in the BuildingConstructiveElementClassValue section of the <a href="#">Building Data Dictionary</a> .
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### Class BuildingConstructiveElementClassValue

Definition: BuildingConstructiveElementClassValue is a code list used to further classify a BuildingConstructiveElement.

Subclass Of: <-->

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.4.8. Class BuildingConstructiveElementFunctionValue

Requirement 88	/req/Building/BuildingConstructiveElementFunctionValue
A	Any use of the BuildingConstructiveElementFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingConstructiveElementFunctionValue UML class as documented in the BuildingConstructiveElementFunctionValue section of the <a href="#">Building Data Dictionary</a> .

#### Class BuildingConstructiveElementFunctionValue

Definition: BuildingConstructiveElementFunctionValue is a code list that enumerates the different purposes of a BuildingConstructiveElement.

Subclass Of: <-->

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.4.9. Class BuildingConstructiveElementUsageValue

Requirement 89	/req/Building/BuildingConstructiveElementUsageValue
A	Any use of the BuildingConstructiveElementUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingConstructiveElementUsageValue UML class as documented in the BuildingConstructiveElementUsageValue section of the <a href="#">Building Data Dictionary</a> .

**Class BuildingConstructiveElementUsageValue**

Definition: BuildingConstructiveElementUsageValue is a code list that enumerates the different uses of a BuildingConstructiveElement.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

**Relations****Attributes****11.4.10. Class BuildingFunctionValue**

<b>Requirement 90</b>	<b>/req/Building/BuildingFunctionValue</b>
A	Any use of the BuildingFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingFunctionValue UML class as documented in the BuildingFunctionValue section of the <a href="#">Building Data Dictionary</a> .

**Class BuildingFunctionValue**

Definition: BuildingFunctionValue is a code list that enumerates the different purposes of a Building.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

**Relations****Attributes****11.4.11. Class BuildingFurniture**

<b>Requirement 91</b>	<b>/req/Building/BuildingFurniture</b>
A	The Implementation Specification SHALL contain an element with the same definition as the BuildingFurniture UML class as documented in the BuildingFurniture section of the <a href="#">Building Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BuildingFurniture UML class as documented in the BuildingFurniture section of the <a href="#">Building Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the BuildingFurniture UML class; including the name, definition, type, and cardinality of those documented in the BuildingFurniture section of the <a href="#">Building Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the BuildingFurniture UML class; including the name, definition, type, and cardinality of those documented in the BuildingFurniture section of the <a href="#">Building Data Dictionary</a> .
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class 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»

### Relations

### Attributes

Attribute Name: class

Value Type: BuildingFurnitureClassValue

Definition: Indicates the specific type of the BuildingFurniture.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: BuildingFurnitureFunctionValue

Definition: Specifies the intended purposes of the BuildingFurniture.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: BuildingFurnitureUsageValue

Definition: Specifies the actual uses of the BuildingFurniture.

Multiplicity: [0..\*]

Stereotype: «Property»

### 11.4.12. Class BuildingFurnitureClassValue

Requirement 92	/req/Building/BuildingFurnitureClassValue
A	Any use of the BuildingFurnitureClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingFurnitureClassValue UML class as documented in the BuildingFurnitureClassValue section of the <a href="#">Building Data Dictionary</a> .

### Class BuildingFurnitureClassValue

Definition: BuildingFurnitureClassValue is a code list used to further classify a BuildingFurniture.

Subclass Of: <-> section,>>

Stereotype: «CodeList»

**Relations**

**Attributes**

### 11.4.13. Class BuildingFurnitureFunctionValue

Requirement 93	/req/Building/BuildingFurnitureFunctionValue
A	Any use of the BuildingFurnitureFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingFurnitureFunctionValue UML class as documented in the BuildingFurnitureFunctionValue section of the <a href="#">Building Data Dictionary</a> .

#### Class BuildingFurnitureFunctionValue

Definition: BuildingFurnitureFunctionValue is a code list that enumerates the different purposes of a BuildingFurniture.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

**Relations**

**Attributes**

### 11.4.14. Class BuildingFurnitureUsageValue

Requirement 94	/req/Building/BuildingFurnitureUsageValue
A	Any use of the BuildingFurnitureUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingFurnitureUsageValue UML class as documented in the BuildingFurnitureUsageValue section of the <a href="#">Building Data Dictionary</a> .

#### Class BuildingFurnitureUsageValue

Definition: BuildingFurnitureUsageValue is a code list that enumerates the different uses of a BuildingFurniture.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

**Relations**

**Attributes**

### 11.4.15. Class BuildingInstallation

Requirement 95 /req/Building/BuildingInstallation

A	The Implementation Specification SHALL contain an element with the same definition as the BuildingInstallation UML class as documented in the BuildingInstallation section of the <a href="#">Building Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BuildingInstallation UML class as documented in the BuildingInstallation section of the <a href="#">Building Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the BuildingInstallation UML class; including the name, definition, type, and cardinality of those documented in the BuildingInstallation section of the <a href="#">Building Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BuildingInstallation UML class; including the name, definition, type, and cardinality of those documented in the BuildingInstallation section of the <a href="#">Building Data Dictionary</a> .

### Class 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»

### Relations

### Attributes

Attribute Name: class

Value Type: BuildingInstallationClassValue

Definition: Indicates the specific type of the BuildingInstallation.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: BuildingInstallationFunctionValue

Definition: Specifies the intended purposes of the BuildingInstallation.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: BuildingInstallationUsageValue

Definition: Specifies the actual uses of the BuildingInstallation.

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.4.16. Class BuildingInstallationClassValue

Requirement 96	/req/Building/BuildingInstallationClassValue
A	Any use of the BuildingInstallationClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingInstallationClassValue UML class as documented in the BuildingInstallationClassValue section of the <a href="#">Building Data Dictionary</a> .

### Class BuildingInstallationClassValue

Definition: BuildingInstallationClassValue is a code list used to further classify a BuildingInstallation.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.4.17. Class BuildingInstallationFunctionValue

Requirement 97	/req/Building/BuildingInstallationFunctionValue
A	Any use of the BuildingInstallationFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingInstallationFunctionValue UML class as documented in the BuildingInstallationFunctionValue section of the <a href="#">Building Data Dictionary</a> .

### Class BuildingInstallationFunctionValue

Definition: BuildingInstallationFunctionValue is a code list that enumerates the different purposes of a BuildingInstallation.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.4.18. Class BuildingInstallationUsageValue

Requirement 98	/req/Building/BuildingInstallationUsageValue
A	Any use of the BuildingInstallationUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingInstallationUsageValue UML class as documented in the BuildingInstallationUsageValue section of the <a href="#">Building Data Dictionary</a> .

### **Class BuildingInstallationUsageValue**

Definition: BuildingInstallationUsageValue is a code list that enumerates the different uses of a BuildingInstallation.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.4.19. Class BuildingPart**

Requirement 99	/req/Building/BuildingPart
A	The Implementation Specification SHALL contain an element with the same definition as the BuildingPart UML class as documented in the BuildingPart section of the <a href="#">Building Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BuildingPart UML class as documented in the BuildingPart section of the <a href="#">Building Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the BuildingPart UML class; including the name, definition, type, and cardinality of those documented in the BuildingPart section of the <a href="#">Building Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BuildingPart UML class; including the name, definition, type, and cardinality of those documented in the BuildingPart section of the <a href="#">Building Data Dictionary</a> .

### **Class 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»

#### **Relations**

#### **Attributes**

### **11.4.20. Class BuildingRoom**

Requirement 100	/req/Building/BuildingRoom
A	The Implementation Specification SHALL contain an element with the same definition as the BuildingRoom UML class as documented in the BuildingRoom section of the <a href="#">Building Data Dictionary</a> .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BuildingRoom UML class as documented in the BuildingRoom section of the <a href="#">Building Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the BuildingRoom UML class; including the name, definition, type, and cardinality of those documented in the BuildingRoom section of the <a href="#">Building Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BuildingRoom UML class; including the name, definition, type, and cardinality of those documented in the BuildingRoom section of the <a href="#">Building Data Dictionary</a> .

## Class 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»

## Relations

Relation Name: buildingFurniture

Cardinality: \*

Target Class: [BuildingFurniture](#)

Definition: Relates the furniture objects to the Building or BuildingPart.

Relation Name: boundary

Cardinality: \*

Target Class: [AbstractThematicSurface](#)

Definition:

Relation Name: buildingInstallation

Cardinality: \*

Target Class: [BuildingInstallation](#)

Definition: Relates to the installation objects.

## Attributes

Attribute Name: class

Value Type: BuildingRoomClassValue

Definition: Indicates the specific type of the BuildingRoom.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: BuildingRoomFunctionValue

Definition: Specifies the intended purposes of the BuildingRoom.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name:	roomHeight
Value Type:	RoomHeight
Definition:	Specifies qualified heights of the BuildingRoom.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BuildingRoomUsageValue
Definition:	Specifies the actual uses of the BuildingRoom.
Multiplicity:	[0..*]
Stereotype:	«Property»

## 11.4.21. Class BuildingRoomClassValue

Requirement 101	/req/Building/BuildingRoomClassValue
A	Any use of the BuildingRoomClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingRoomClassValue UML class as documented in the BuildingRoomClassValue section of the <a href="#">Building Data Dictionary</a> .

### Class BuildingRoomClassValue

Definition: BuildingRoomClassValue is a code list used to further classify a BuildingRoom.  
 Subclass Of: <-- section,>>  
 Stereotype: «CodeList»

### Relations

### Attributes

## 11.4.22. Class BuildingRoomFunctionValue

Requirement 102	/req/Building/BuildingRoomFunctionValue
A	Any use of the BuildingRoomFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingRoomFunctionValue UML class as documented in the BuildingRoomFunctionValue section of the <a href="#">Building Data Dictionary</a> .

### Class BuildingRoomFunctionValue

Definition: BuildingRoomFunctionValue is a code list that enumerates the different purposes of a BuildingRoom.  
 Subclass Of: <-- section,>>  
 Stereotype: «CodeList»

### Relations

### Attributes

### 11.4.23. Class BuildingRoomUsageValue

<b>Requirement 103</b>	/req/Building/BuildingRoomUsageValue
A	Any use of the BuildingRoomUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingRoomUsageValue UML class as documented in the BuildingRoomUsageValue section of the <a href="#">Building Data Dictionary</a> .

#### Class BuildingRoomUsageValue

Definition: BuildingRoomUsageValue is a code list that enumerates the different uses of a BuildingRoom.

Subclass Of: <-- section,-->

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.4.24. Class BuildingSubdivisionClassValue

<b>Requirement 104</b>	/req/Building/BuildingSubdivisionClassValue
A	Any use of the BuildingSubdivisionClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingSubdivisionClassValue UML class as documented in the BuildingSubdivisionClassValue section of the <a href="#">Building Data Dictionary</a> .

#### Class BuildingSubdivisionClassValue

Definition: BuildingSubdivisionClassValue is a code list used to further classify a BuildingSubdivision.

Subclass Of: <-- section,-->

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.4.25. Class BuildingSubdivisionFunctionValue

<b>Requirement 105</b>	/req/Building/BuildingSubdivisionFunctionValue
------------------------	------------------------------------------------

A	Any use of the BuildingSubdivisionFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingSubdivisionFunctionValue UML class as documented in the BuildingSubdivisionFunctionValue section of the <a href="#">Building Data Dictionary</a> .
---	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class BuildingSubdivisionFunctionValue

Definition: BuildingSubdivisionFunctionValue is a code list that enumerates the different purposes of a BuildingSubdivision.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.4.26. Class BuildingSubdivisionUsageValue

Requirement 106	/req/Building/BuildingSubdivisionUsageValue
A	Any use of the BuildingSubdivisionUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingSubdivisionUsageValue UML class as documented in the BuildingSubdivisionUsageValue section of the <a href="#">Building Data Dictionary</a> .

### Class BuildingSubdivisionUsageValue

Definition: BuildingSubdivisionUsageValue is a code list that enumerates the different uses of a BuildingSubdivision.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.4.27. Class BuildingUnit

Requirement 107	/req/Building/BuildingUnit
A	The Implementation Specification SHALL contain an element with the same definition as the BuildingUnit UML class as documented in the BuildingUnit section of the <a href="#">Building Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BuildingUnit UML class as documented in the BuildingUnit section of the <a href="#">Building Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the BuildingUnit UML class; including the name, definition, type, and cardinality of those documented in the BuildingUnit section of the <a href="#">Building Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BuildingUnit UML class; including the name, definition, type, and cardinality of those documented in the BuildingUnit section of the <a href="#">Building Data Dictionary</a> .

### Class BuildingUnit

Definition: A BuildingUnit is a logical subdivision of a Building. BuildingUnits are formed according to some homogeneous property like function, ownership, management, or accessibility. They may be separately sold, rented out, inherited, managed, etc.

Subclass Of: [AbstractBuildingSubdivision](#)

Stereotype: «FeatureType»

### Relations

Relation Name: address

Cardinality: \*

Target Class: [Address](#)

Definition: SIG3D: Relation between BuildingUnit and Address.

### Attributes

## 11.4.28. Class BuildingUsageValue

Requirement 108	/req/Building/BuildingUsageValue
A	Any use of the BuildingUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingUsageValue UML class as documented in the BuildingUsageValue section of the <a href="#">Building Data Dictionary</a> .

### Class BuildingUsageValue

Definition: BuildingUsageValue is a code list that enumerates the different uses of a Building.

Subclass Of: <-> section,>>

Stereotype: «CodeList»

### Relations

### Attributes

## 11.4.29. Class RoofTypeValue

Requirement 109	/req/Building/RoofTypeValue
-----------------	-----------------------------

A	Any use of the RoofTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RoofTypeValue UML class as documented in the RoofTypeValue section of the <a href="#">Building Data Dictionary</a> .
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class RoofTypeValue

Definition: RoofTypeValue is a code list that enumerates different roof types.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.4.30. Class RoomElevationReferenceValue

Requirement 110	/req/Building/RoomElevationReferenceValue
A	Any use of the RoomElevationReferenceValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RoomElevationReferenceValue UML class as documented in the RoomElevationReferenceValue section of the <a href="#">Building Data Dictionary</a> .

### Class RoomElevationReferenceValue

Definition: RoomElevationReferenceValue is a code list that enumerates the different elevation reference levels used to measure room heights.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.4.31. Class Storey

Requirement 111	/req/Building/Storey
A	The Implementation Specification SHALL contain an element with the same definition as the Storey UML class as documented in the Storey section of the <a href="#">Building Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Storey UML class as documented in the Storey section of the <a href="#">Building Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the Storey UML class; including the name, definition, type, and cardinality of those documented in the Storey section of the <a href="#">Building Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Storey UML class; including the name, definition, type, and cardinality of those documented in the Storey section of the <a href="#">Building Data Dictionary</a> .

### Class Storey

Definition: A Storey is a horizontal section of a Building.

Subclass Of: [AbstractBuildingSubdivision](#)

Stereotype: «FeatureType»

### Relations

Relation Name: buildingUnit

Cardinality: \*

Target Class: [BuildingUnit](#)

Definition: Relates to the building units that belong to the Storey.

Relation Name: boundary

Cardinality: \*

Target Class: [AbstractThematicSurface](#)

Definition:

### Attributes

## 11.4.32. Class RoomHeight

Requirement 112	/req/Building/RoomHeight
A	Any use of the RoomHeight type in the Implementation Specification SHALL have the same definition as the RoomHeight UML class as documented in the RoomHeight section of the <a href="#">Building Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the RoomHeight UML class as documented in the RoomHeight section of the <a href="#">Building Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the RoomHeight UML class; including the name, definition, type, and cardinality of those documented in the RoomHeight section of the <a href="#">Building Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the RoomHeight UML class; including the name, definition, type, and cardinality of those documented in the RoomHeight section of the <a href="#">Building Data Dictionary</a> .

## **Class RoomHeight**

Definition: The RoomHeight represents a vertical distance (measured or estimated) between a low reference and a high reference. [cf. INSPIRE]

Subclass Of: <-- section,>>

Stereotype: «DataType»

## **Relations**

## **Attributes**

Attribute Name: highReference

Value Type: RoomElevationReferenceValue

Definition: Indicates the high point used to calculate the value of the room height.

Multiplicity:

Stereotype: «Property»

Attribute Name: lowReference

Value Type: RoomElevationReferenceValue

Definition: Indicates the low point used to calculate the value of the room height.

Multiplicity:

Stereotype: «Property»

Attribute Name: status

Value Type: HeightStatusValue

Definition: Indicates the way the room height has been captured.

Multiplicity:

Stereotype: «Property»

Attribute Name: value

Value Type: Length

Definition: Specifies the value of the room height.

Multiplicity:

Stereotype: «Property»

## **11.4.33. Additional Information**

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

## 11.5. City Furniture

### Requirements Class

<http://www.opengis.net/spec/CityGML/3.0/req/req-class-cityfurniture>

Target type	Conceptual Model
Dependency	<a href="#">/req/req-class-core</a>

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](#).

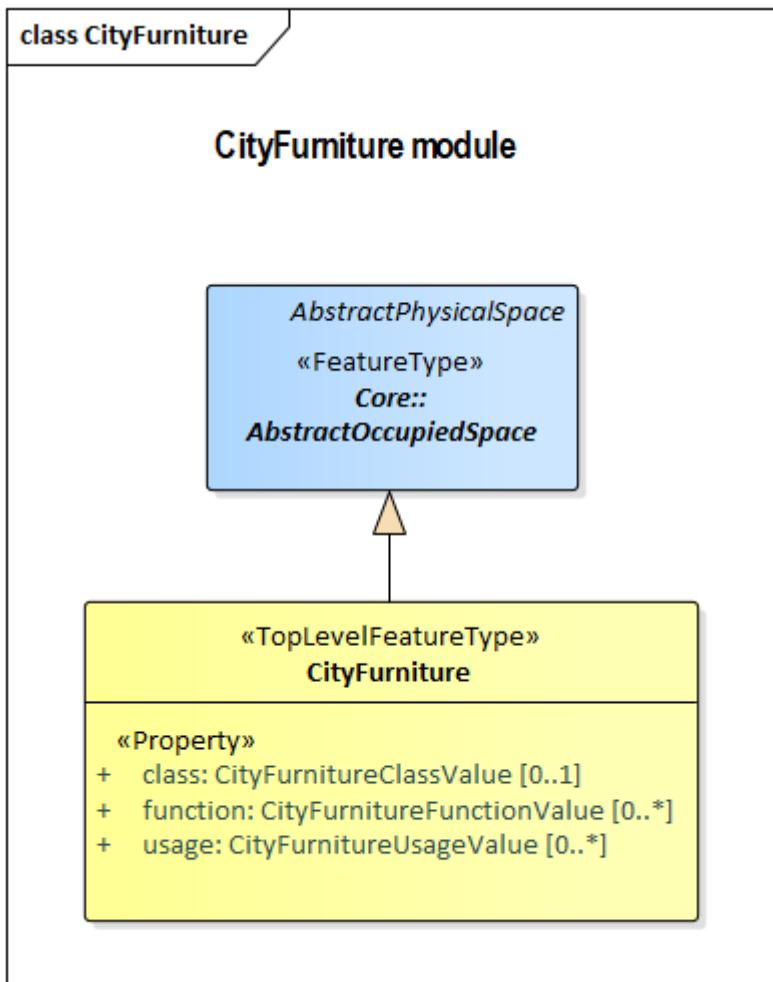


Figure 18. UML diagram of CityGML's city furniture model.

### 11.5.1. CityFurniture 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.

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.5.2. Class CityFurniture

Requirement 113	/req/CityFurniture/CityFurniture
A	The Implementation Specification SHALL contain an element with the same definition as the CityFurniture UML class as documented in the CityFurniture section of the <a href="#">CityFurniture Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the CityFurniture UML class as documented in the CityFurniture section of the <a href="#">CityFurniture Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the CityFurniture UML class; including the name, definition, type, and cardinality of those documented in the CityFurniture section of the <a href="#">CityFurniture Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the CityFurniture UML class; including the name, definition, type, and cardinality of those documented in the CityFurniture section of the <a href="#">CityFurniture Data Dictionary</a> .

#### Class 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»

#### Relations

#### Attributes

Attribute Name: class

Value Type: CityFurnitureClassValue

Definition: Indicates the specific type of the CityFurniture.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	function
Value Type:	CityFurnitureFunctionValue
Definition:	Specifies the intended purposes of the CityFurniture.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	CityFurnitureUsageValue
Definition:	Specifies the actual uses of the CityFurniture.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.5.3. Class CityFurnitureClassValue

Requirement 114	/req/CityFurniture/CityFurnitureClassValue
A	Any use of the CityFurnitureClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the CityFurnitureClassValue UML class as documented in the <a href="#">CityFurniture Data Dictionary</a> .

#### Class CityFurnitureClassValue

Definition: CityFurnitureClassValue is a code list used to further classify a CityFurniture.  
 Subclass Of: <-- section,>>  
 Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.5.4. Class CityFurnitureFunctionValue

Requirement 115	/req/CityFurniture/CityFurnitureFunctionValue
A	Any use of the CityFurnitureFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the CityFurnitureFunctionValue UML class as documented in the <a href="#">CityFurniture Data Dictionary</a> .

#### Class CityFurnitureFunctionValue

Definition: CityFurnitureFunctionValue is a code list that enumerates the different purposes of a CityFurniture.  
 Subclass Of: <-- section,>>  
 Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.5.5. Class CityFurnitureUsageValue

Requirement 116	/req/CityFurniture/CityFurnitureUsageValue
A	Any use of the CityFurnitureUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the CityFurnitureUsageValue UML class as documented in the CityFurnitureUsageValue section of the <a href="#">CityFurniture Data Dictionary</a> .

### Class CityFurnitureUsageValue

Definition: CityFurnitureUsageValue is a code list that enumerates the different uses of a CityFurniture.

Subclass Of: <-- section,-->

Stereotype: «CodeList»

### Relations

### Attributes

## 11.5.6. Additional Information

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

## 11.6. Module City Furniture new 1

### Requirements Class

<http://www.opengis.net/spec/CityGML/3.0/req/req-class-cityfurniture>

Target type	Conceptual Model
Dependency	<a href="#">/req/req-class-core</a>

### 11.6.1. Introduction and UML model

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](#).

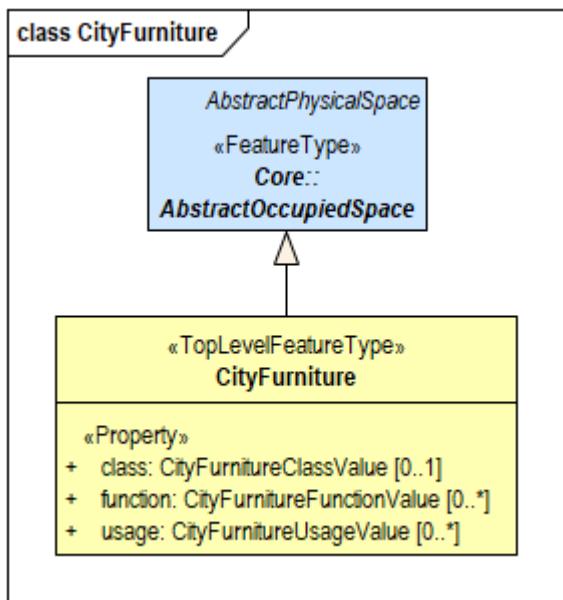


Figure 19. UML diagram of CityGML's city furniture model.

### Requirement /req/CityFurniture/CityFurniture

The CityFurniture Requirements Class classes shown in yellow in Figure [City Furniture UML Diagram](#) shall be provided by the encoding in a manner consistent with the encoding.

### 11.6.2. Package

#### Package CityFurniture

Definition:	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»

## 11.6.3. Classes

### 11.6.3.1. 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:	<a href="#">AbstractOccupiedSpace</a>
Stereotype:	«TopLevelFeatureType»

#### Relations

**Attributes** (Unless otherwise specified, all attributes have the stereotype «Property».)

##### **Attribute: class**

Name:	class
Value type:	<a href="#">CityFurnitureClassValue</a>
Definition:	Indicates the specific type of the CityFurniture.
Multiplicity:	[0..1]

##### **Attribute: function**

Name:	function
Value type:	<a href="#">CityFurnitureFunctionValue</a>
Definition:	Specifies the intended purposes of the CityFurniture.
Multiplicity:	[0..*]

### **Attribute: usage**

Name: function  
Value type: [CityFurnitureUsageValue](#)  
Definition: Specifies the actual uses of the CityFurniture.  
Multiplicity: [0..\*]

## **11.6.4. Code lists**

### **11.6.4.1. Code list CityFurnitureClassValue**

#### **Code list CityFurnitureClassValue**

Definition: CityFurnitureClassValue is a code list used to further classify a CityFurniture.  
Subclass of: -  
Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.6.4.2. Code list CityFurnitureFunctionValue**

#### **Code list CityFurnitureFunctionValue**

Definition: CityFurnitureFunctionValue is a code list that enumerates the different purposes of a CityFurniture.  
Subclass of: -  
Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.6.4.3. Code list CityFurnitureUsageValue**

#### **Code list CityFurnitureUsageValue**

Definition: CityFurnitureUsageValue is a code list that enumerates the different uses of a CityFurniture.

Subclass of: -

Stereotype: «CodeList»

## Relations

## Attributes

### 11.6.5. Additional Information

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

## 11.7. Module City Furniture new 2

### Requirements Class

<http://www.opengis.net/spec/CityGML/3.0/req/req-class-cityfurniture>

Target type	Conceptual Model
Dependency	<a href="#">/req/req-class-core</a>

### 11.7.1. Introduction and UML model

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](#).

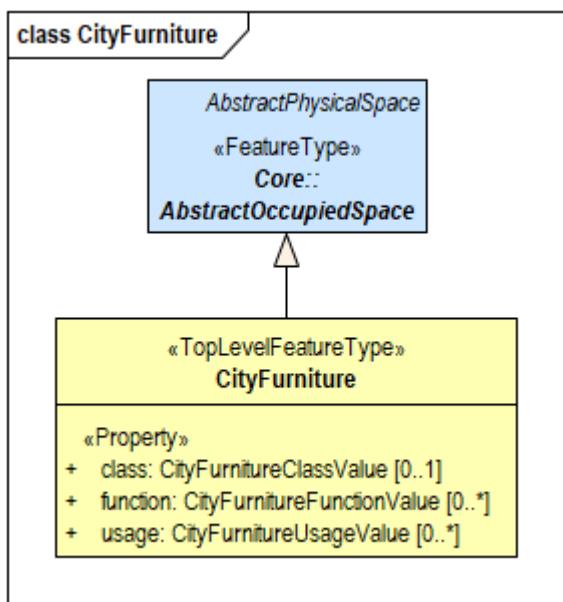


Figure 20. UML diagram of CityGML's city furniture model.

### Requirement /req/CityFurniture/CityFurniture

The CityFurniture Requirements Class classes shown in yellow in Figure [City Furniture UML Diagram](#) shall be provided by the encoding in a manner consistent with the encoding.

### 11.7.2. Package

#### Package CityFurniture

Definition: 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»

## 11.7.3. Classes

### 11.7.3.1. 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»

#### Attributes

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

Attribute	Value type and multiplicity	Definition
class	<a href="#">CityFurnitureClassValue</a> [0..1]	Indicates the specific type of the CityFurniture.
function	<a href="#">CityFurnitureFunctionValue</a> [0..*]	Specifies the intended purposes of the CityFurniture.
usage	<a href="#">CityFurnitureUsageValue</a> [0..*]	Specifies the actual uses of the CityFurniture.

#### Relations

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

Role name	Target class and multiplicity	Definition
-	-	-

## 11.7.4. Code lists

All code lists have the stereotype «CodeList».

<b>Code list</b>	<b>Subclass of</b>	<b>Stereotype</b>
CityFurnitureClassValue	-	CityFurnitureClassValue is a code list used to further classify a CityFurniture.
CityFurnitureFunctionValue	-	CityFurnitureFunctionValue is a code list that enumerates the different purposes of a CityFurniture.
CityFurnitureUsageValue	-	CityFurnitureUsageValue is a code list that enumerates the different uses of a CityFurniture.

### 11.7.5. Additional Information

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

## 11.8. City Object Group

### Requirements Class

<http://www.opengis.net/spec/CityGML/3.0/req/req-class-cityobjectgroup>

Target type	Conceptual Model
Dependency	<a href="#">/req/req-class-core</a>

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](#).

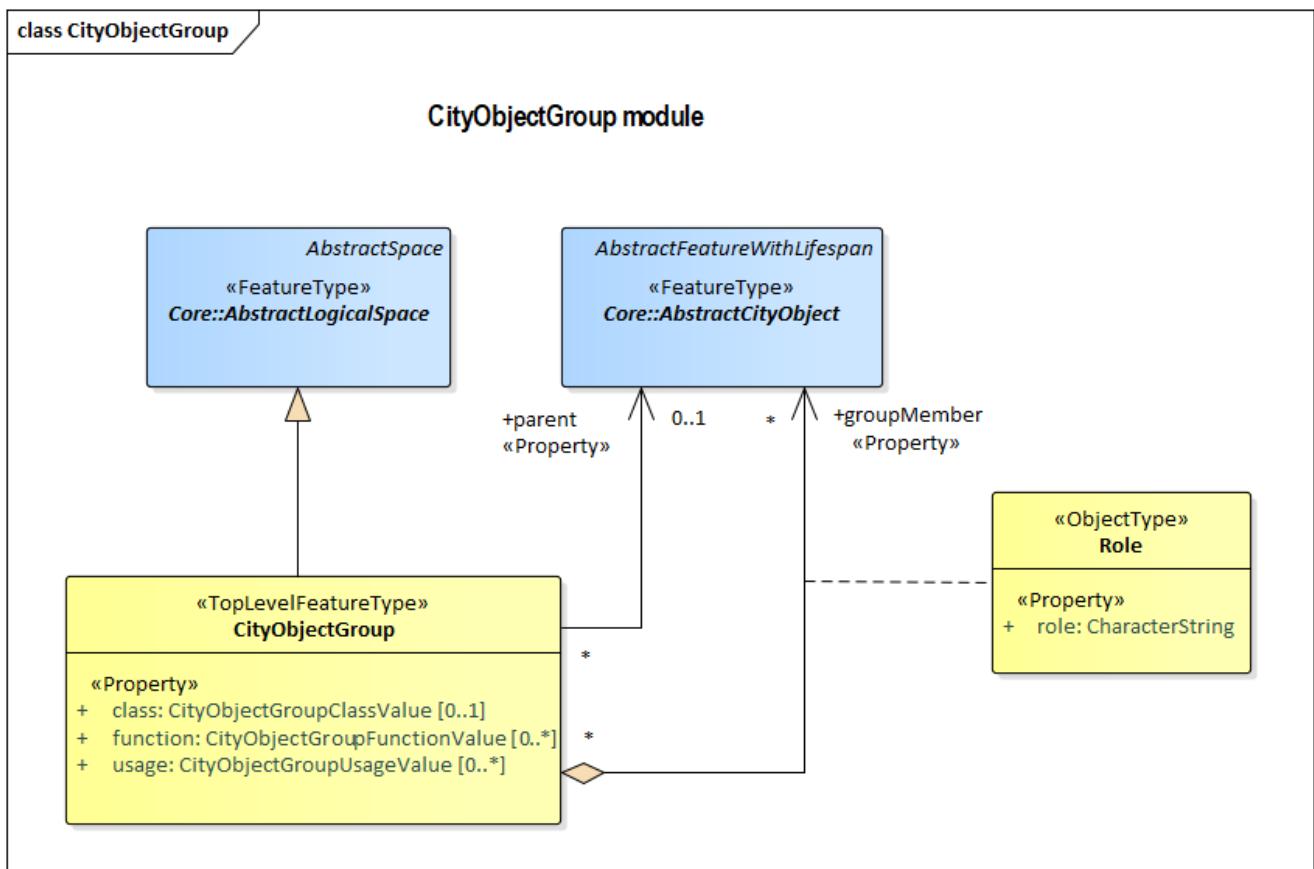


Figure 21. UML diagram of the City Object Group Model.

## 11.8.1. CityObjectGroup 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.

Stereotype: «ApplicationSchema»

Parent Package: CityGML

## 11.8.2. Class CityObjectGroup

Requirement 117	/req/CityObjectGroup/CityObjectGroup
A	The Implementation Specification SHALL contain an element with the same definition as the CityObjectGroup UML class as documented in the CityObjectGroup section of the <a href="#">CityObjectGroup Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the CityObjectGroup UML class as documented in the CityObjectGroup section of the <a href="#">CityObjectGroup Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the CityObjectGroup UML class; including the name, definition, type, and cardinality of those documented in the CityObjectGroup section of the <a href="#">CityObjectGroup Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the CityObjectGroup UML class; including the name, definition, type, and cardinality of those documented in the CityObjectGroup section of the <a href="#">CityObjectGroup Data Dictionary</a> .

### Class 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»

### Relations

Relation Name: parent

Cardinality: 0..1

Target Class: [AbstractCityObject](#)

Definition: Relates to a city object to which the CityObjectGroup belongs.

Relation Name:	groupMember
Cardinality:	*
Target Class:	<a href="#">AbstractCityObject</a>
Definition:	Relates to the city objects that are part of the CityObjectGroup.
<b>Attributes</b>	
Attribute Name:	class
Value Type:	CityObjectGroupClassValue
Definition:	Indicates the specific type of the CityObjectGroup.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	CityObjectGroupFunctionValue
Definition:	Specifies the intended purposes of the CityObjectGroup.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	CityObjectGroupUsageValue
Definition:	Specifies the actual usages of the CityObjectGroup.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.8.3. Class CityObjectGroupClassValue

Requirement 118	/req/CityObjectGroup/CityObjectGroupClassValue
A	Any use of the CityObjectGroupClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the CityObjectGroupClassValue UML class as documented in the CityObjectGroupClassValue section of the <a href="#">CityObjectGroup Data Dictionary</a> .

#### Class CityObjectGroupClassValue

Definition: CityObjectGroupClassValue is a code list used to further classify a CityObjectGroup.  
 Subclass Of: <-- section,>>  
 StereoType: «CodeList»

#### Relations

#### Attributes

### 11.8.4. Class CityObjectGroupFunctionValue

Requirement 119	/req/CityObjectGroup/CityObjectGroupFunctionValue
-----------------	---------------------------------------------------

A	Any use of the CityObjectGroupFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the CityObjectGroupFunctionValue UML class as documented in the CityObjectGroupFunctionValue section of the <a href="#">CityObjectGroup Data Dictionary</a> .
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class CityObjectGroupFunctionValue

Definition: CityObjectGroupFunctionValue is a code list that enumerates the different purposes of a CityObjectGroup.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.8.5. Class CityObjectGroupUsageValue

Requirement 120	/req/CityObjectGroup/CityObjectGroupUsageValue
A	Any use of the CityObjectGroupUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the CityObjectGroupUsageValue UML class as documented in the CityObjectGroupUsageValue section of the <a href="#">CityObjectGroup Data Dictionary</a> .

### Class CityObjectGroupUsageValue

Definition: CityObjectGroupUsageValue is a code list that enumerates the different uses of a CityObjectGroup.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.8.6. Class Role

Requirement 121	/req/CityObjectGroup/Role
A	The Implementation Specification SHALL contain an element with the same definition as the Role UML class as documented in the Role section of the <a href="#">CityObjectGroup Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Role UML class as documented in the Role section of the <a href="#">CityObjectGroup Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the Role UML class; including the name, definition, type, and cardinality of those documented in the Role section of the <a href="#">CityObjectGroup Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Role UML class; including the name, definition, type, and cardinality of those documented in the Role section of the <a href="#">CityObjectGroup Data Dictionary</a> .

### Class Role

Definition: Role qualifies the function of a city object within the CityObjectGroup.

Subclass Of: <-- section,>>

Stereotype: «ObjectType»

### Relations

### Attributes

Attribute Name: role

Value Type: CharacterString

Definition: Describes the role the city object plays within the CityObjectGroup.

Multiplicity:

Stereotype: «Property»

## 11.8.7. Additional Information

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

## 11.9. Construction

### Requirements Class

<http://www.opengis.net/spec/CityGML/3.0/req/req-class-construction>

Target type	Conceptual Model
Dependency	<a href="/req/req-class-core">/req/req-class-core</a>

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 Construction 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](#).

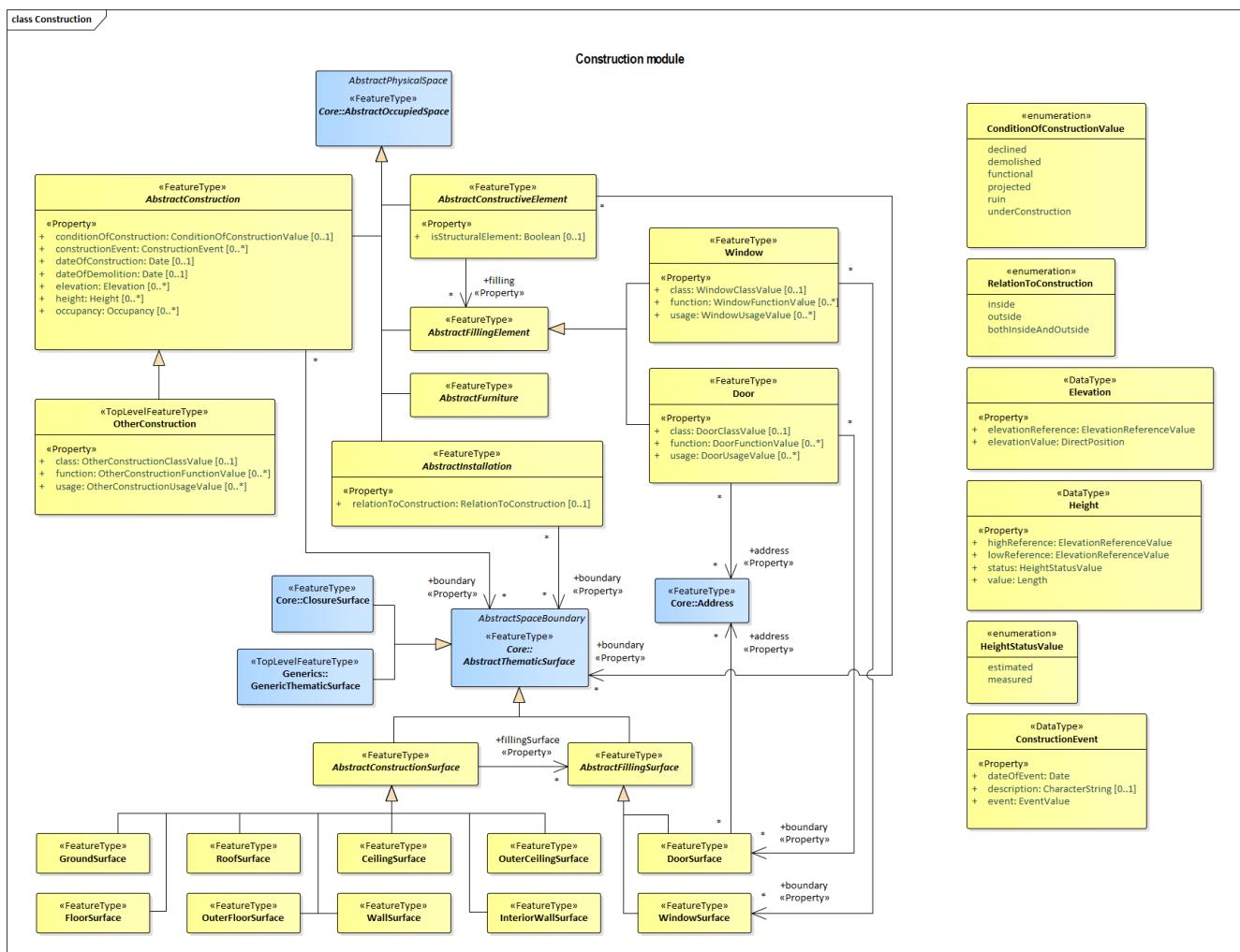


Figure 22. UML diagram of the Construction Model.

### 11.9.1. Construction 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.
Stereotype:	«ApplicationSchema»
Parent Package:	CityGML

## 11.9.2. Class AbstractConstruction

Requirement 122	/req/Construction/AbstractConstruction
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractConstruction UML class as documented in the AbstractConstruction section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractConstruction UML class as documented in the AbstractConstruction section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractConstruction UML class; including the name, definition, type, and cardinality of those documented in the AbstractConstruction section of the <a href="#">Construction Data Dictionary</a> .

### Class 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»

### Relations

Relation Name: boundary

Cardinality: \*

Target Class: [AbstractThematicSurface](#)

Definition:

### Attributes

Attribute Name: conditionOfConstruction

Value Type: ConditionOfConstructionValue

Definition: Indicates the life-cycle status of the construction. [cf. INSPIRE]

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	constructionEvent
Value Type:	ConstructionEvent
Definition:	Describes specific events in the life-time of the construction.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	dateOfConstruction
Value Type:	Date
Definition:	Indicates the date at which the construction was completed.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	dateOfDemolition
Value Type:	Date
Definition:	Indicates the date at which the construction was demolished.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	elevation
Value Type:	Elevation
Definition:	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]
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	height
Value Type:	Height
Definition:	Specifies qualified heights of the construction above ground or below ground. [cf. INSPIRE]
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	occupancy
Value Type:	Occupancy
Definition:	Provides qualified information on the residency of persons, animals, or other moveable objects in the construction.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.9.3. Class AbstractConstructionSurface

Requirement 123	/req/Construction/AbstractConstructionSurface
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractConstructionSurface UML class as documented in the AbstractConstructionSurface section of the <a href="#">Construction Data Dictionary</a> .

B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractConstructionSurface UML class as documented in the AbstractConstructionSurface section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractConstructionSurface UML class; including the name, definition, type, and cardinality of those documented in the AbstractConstructionSurface section of the <a href="#">Construction Data Dictionary</a> .

#### Class AbstractConstructionSurface

Definition: AbstractConstructionSurface is the abstract superclass for different kinds of surfaces that bound a construction.

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

#### Relations

Relation Name: fillingSurface

Cardinality: \*

Target Class: [AbstractFillingSurface](#)

Definition: Relates to the surfaces that seal the openings of the construction surface.

#### Attributes

#### 11.9.4. Class AbstractConstructiveElement

Requirement 124	/req/Construction/AbstractConstructiveElement
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractConstructiveElement UML class as documented in the AbstractConstructiveElement section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractConstructiveElement UML class as documented in the AbstractConstructiveElement section of the <a href="#">Construction Data Dictionary</a> .

C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the AbstractConstructiveElement section of the <a href="#">Construction Data Dictionary</a> .
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class 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»

#### Relations

Relation Name: filling

Cardinality: \*

Target Class: [AbstractFillingElement](#)

Definition: Relates to the elements that fill the opening of the constructive element.

Relation Name: boundary

Cardinality: \*

Target Class: [AbstractThematicSurface](#)

Definition:

#### Attributes

Attribute Name: isStructuralElement

Value Type: Boolean

Definition: 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.

Multiplicity: [0..1]

Stereotype: «Property»

### 11.9.5. Class AbstractFillingElement

Requirement 125	/req/Construction/AbstractFillingElement
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractFillingElement UML class as documented in the AbstractFillingElement section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractFillingElement UML class as documented in the AbstractFillingElement section of the <a href="#">Construction Data Dictionary</a> .

C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractFillingElement UML class; including the name, definition, type, and cardinality of those documented in the AbstractFillingElement section of the <a href="#">Construction Data Dictionary</a> .
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class AbstractFillingElement

Definition: AbstractFillingElement is the abstract superclass for different kinds of elements that fill the openings of constructive elements.

Subclass Of: [AbstractOccupiedSpace](#)

Stereotype: «FeatureType»

#### Relations

#### Attributes

### 11.9.6. Class AbstractFillingSurface

Requirement 126	/req/Construction/AbstractFillingSurface
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractFillingSurface UML class as documented in the AbstractFillingSurface section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractFillingSurface UML class as documented in the AbstractFillingSurface section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractFillingSurface UML class; including the name, definition, type, and cardinality of those documented in the AbstractFillingSurface section of the <a href="#">Construction Data Dictionary</a> .

### Class AbstractFillingSurface

Definition: AbstractFillingSurface is the abstract superclass for different kinds of surfaces that seal openings filled by filling elements.

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

#### Relations

#### Attributes

### 11.9.7. Class AbstractFurniture

Requirement 127	/req/Construction/AbstractFurniture
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractFurniture UML class as documented in the AbstractFurniture section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractFurniture UML class as documented in the AbstractFurniture section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractFurniture UML class; including the name, definition, type, and cardinality of those documented in the AbstractFurniture section of the <a href="#">Construction Data Dictionary</a> .

#### Class AbstractFurniture

Definition: AbstractFurniture is the abstract superclass for the representation of furniture objects of a construction.

Subclass Of: [AbstractOccupiedSpace](#)

Stereotype: «FeatureType»

#### Relations

#### Attributes

### 11.9.8. Class AbstractInstallation

Requirement 128	/req/Construction/AbstractInstallation
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractInstallation UML class as documented in the AbstractInstallation section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractInstallation UML class as documented in the AbstractInstallation section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractInstallation UML class; including the name, definition, type, and cardinality of those documented in the AbstractInstallation section of the <a href="#">Construction Data Dictionary</a> .

## Class AbstractInstallation

Definition: AbstractInstallation is the abstract superclass for the representation of installation objects of a construction.

Subclass Of: [AbstractOccupiedSpace](#)

Stereotype: «FeatureType»

## Relations

Relation Name: boundary

Cardinality: \*

Target Class: [AbstractThematicSurface](#)

Definition:

## Attributes

Attribute Name: relationToConstruction

Value Type: RelationToConstruction

Definition: Indicates whether the installation is located inside and/or outside of the construction.

Multiplicity: [0..1]

Stereotype: «Property»

## 11.9.9. Class CeilingSurface

Requirement 129	/req/Construction/CeilingSurface
A	The Implementation Specification SHALL contain an element with the same definition as the CeilingSurface UML class as documented in the CeilingSurface section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the CeilingSurface UML class as documented in the CeilingSurface section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the CeilingSurface UML class; including the name, definition, type, and cardinality of those documented in the CeilingSurface section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the CeilingSurface UML class; including the name, definition, type, and cardinality of those documented in the CeilingSurface section of the <a href="#">Construction Data Dictionary</a> .

## Class 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»

**Relations****Attributes****11.9.10. Class Door**

<b>Requirement 130</b>	/req/Construction/Door
A	The Implementation Specification SHALL contain an element with the same definition as the Door UML class as documented in the Door section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Door UML class as documented in the Door section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Door UML class; including the name, definition, type, and cardinality of those documented in the Door section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Door UML class; including the name, definition, type, and cardinality of those documented in the Door section of the <a href="#">Construction Data Dictionary</a> .

**Class 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»

**Relations**

Relation Name: boundary

Cardinality: \*

Target Class: [DoorSurface](#)

Definition:

Relation Name: address

Cardinality: \*

Target Class: [Address](#)

Definition: Relates to the addresses that are assigned to the Door.

**Attributes**

Attribute Name: class

Value Type: [DoorClassValue](#)

Definition: Indicates the specific type of the Door.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	function
Value Type:	DoorFunctionValue
Definition:	Specifies the intended purposes of the Door.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	DoorUsageValue
Definition:	Specifies the actual uses of the Door.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.9.11. Class DoorClassValue

Requirement 131	/req/Construction/DoorClassValue
A	Any use of the DoorClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the DoorClassValue UML class as documented in the <a href="#">Construction Data Dictionary</a> .

#### Class DoorClassValue

Definition: DoorClassValue is a code list used to further classify a Door.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.9.12. Class DoorFunctionValue

Requirement 132	/req/Construction/DoorFunctionValue
A	Any use of the DoorFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the DoorFunctionValue UML class as documented in the <a href="#">Construction Data Dictionary</a> .

#### Class DoorFunctionValue

Definition: DoorFunctionValue is a code list that enumerates the different purposes of a Door.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.9.13. Class DoorSurface

Requirement 133	/req/Construction/DoorSurface
A	The Implementation Specification SHALL contain an element with the same definition as the DoorSurface UML class as documented in the DoorSurface section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the DoorSurface UML class as documented in the DoorSurface section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the DoorSurface UML class; including the name, definition, type, and cardinality of those documented in the DoorSurface section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the DoorSurface UML class; including the name, definition, type, and cardinality of those documented in the DoorSurface section of the <a href="#">Construction Data Dictionary</a> .

#### Class 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»

#### Relations

Relation Name: address

Cardinality: \*

Target Class: [Address](#)

Definition: Relates to the addresses that are assigned to the DoorSurface.

#### Attributes

### 11.9.14. Class DoorUsageValue

Requirement 134	/req/Construction/DoorUsageValue
A	Any use of the DoorUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the DoorUsageValue UML class as documented in the DoorUsageValue section of the <a href="#">Construction Data Dictionary</a> .

#### Class DoorUsageValue

Definition: DoorUsageValue is a code list that enumerates the different uses of a Door.

Subclass Of: <– section,>>

Stereotype: «CodeList»

**Relations****Attributes**

### 11.9.15. Class ElevationReferenceValue

Requirement 135	/req/Construction/ElevationReferenceValue
A	Any use of the ElevationReferenceValue type in an Implementation Specification SHALL be restricted to the valid values specified in the ElevationReferenceValue UML class as documented in the ElevationReferenceValue section of the <a href="#">Construction Data Dictionary</a> .

**Class ElevationReferenceValue**

Definition: ElevationReferenceValue is a code list that enumerates the different elevation reference levels used to measure construction heights.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

**Relations****Attributes**

### 11.9.16. Class EventValue

Requirement 136	/req/Construction/EventValue
A	Any use of the EventValue type in an Implementation Specification SHALL be restricted to the valid values specified in the EventValue UML class as documented in the EventValue section of the <a href="#">Construction Data Dictionary</a> .

**Class EventValue**

Definition: EventValue is a code list that enumerates the different events of a construction.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

**Relations****Attributes**

### 11.9.17. Class FloorSurface

Requirement 137	/req/Construction/FloorSurface
A	The Implementation Specification SHALL contain an element with the same definition as the FloorSurface UML class as documented in the FloorSurface section of the <a href="#">Construction Data Dictionary</a> .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the FloorSurface UML class as documented in the FloorSurface section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the FloorSurface UML class; including the name, definition, type, and cardinality of those documented in the FloorSurface section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the FloorSurface UML class; including the name, definition, type, and cardinality of those documented in the FloorSurface section of the <a href="#">Construction Data Dictionary</a> .

### Class 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»

#### Relations

#### Attributes

### 11.9.18. Class GroundSurface

Requirement 138	/req/Construction/GroundSurface
A	The Implementation Specification SHALL contain an element with the same definition as the GroundSurface UML class as documented in the GroundSurface section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GroundSurface UML class as documented in the GroundSurface section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the GroundSurface UML class; including the name, definition, type, and cardinality of those documented in the GroundSurface section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GroundSurface UML class; including the name, definition, type, and cardinality of those documented in the GroundSurface section of the <a href="#">Construction Data Dictionary</a> .

### **Class 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»

### **Relations**

### **Attributes**

## **11.9.19. Class InteriorWallSurface**

Requirement 139	/req/Construction/InteriorWallSurface
A	The Implementation Specification SHALL contain an element with the same definition as the InteriorWallSurface UML class as documented in the InteriorWallSurface section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the InteriorWallSurface UML class as documented in the InteriorWallSurface section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the InteriorWallSurface UML class; including the name, definition, type, and cardinality of those documented in the InteriorWallSurface section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the InteriorWallSurface UML class; including the name, definition, type, and cardinality of those documented in the InteriorWallSurface section of the <a href="#">Construction Data Dictionary</a> .

### **Class 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»

### **Relations**

### **Attributes**

## **11.9.20. Class OtherConstruction**

Requirement 140	/req/Construction/OtherConstruction
-----------------	-------------------------------------

A	The Implementation Specification SHALL contain an element with the same definition as the OtherConstruction UML class as documented in the OtherConstruction section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the OtherConstruction UML class as documented in the OtherConstruction section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the OtherConstruction UML class; including the name, definition, type, and cardinality of those documented in the OtherConstruction section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the OtherConstruction UML class; including the name, definition, type, and cardinality of those documented in the OtherConstruction section of the <a href="#">Construction Data Dictionary</a> .

### Class OtherConstruction

Definition: An OtherConstruction is a construction that is not covered by any of the other subclasses of AbstractConstruction.

Subclass Of: [AbstractConstruction](#)

Stereotype: «TopLevelFeatureType»

### Relations

### Attributes

Attribute Name: class

Value Type: OtherConstructionClassValue

Definition: Indicates the specific type of the OtherConstruction.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: OtherConstructionFunctionValue

Definition: Specifies the intended purposes of the OtherConstruction.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: OtherConstructionUsageValue

Definition: Specifies the actual uses of the OtherConstruction.

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.9.21. Class OtherConstructionClassValue

Requirement 141	/req/Construction/OtherConstructionClassValue
A	Any use of the OtherConstructionClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the OtherConstructionClassValue UML class as documented in the OtherConstructionClassValue section of the <a href="#">Construction Data Dictionary</a> .

### Class OtherConstructionClassValue

Definition: OtherConstructionClassValue is a code list used to further classify an OtherConstruction.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.9.22. Class OtherConstructionFunctionValue

Requirement 142	/req/Construction/OtherConstructionFunctionValue
A	Any use of the OtherConstructionFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the OtherConstructionFunctionValue UML class as documented in the OtherConstructionFunctionValue section of the <a href="#">Construction Data Dictionary</a> .

### Class OtherConstructionFunctionValue

Definition: OtherConstructionFunctionValue is a code list that enumerates the different purposes of an OtherConstruction.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.9.23. Class OtherConstructionUsageValue

Requirement 143	/req/Construction/OtherConstructionUsageValue
A	Any use of the OtherConstructionUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the OtherConstructionUsageValue UML class as documented in the OtherConstructionUsageValue section of the <a href="#">Construction Data Dictionary</a> .

### **Class OtherConstructionUsageValue**

Definition: OtherConstructionUsageValue is a code list that enumerates the different uses of an OtherConstruction.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.9.24. Class OuterCeilingSurface**

<b>Requirement 144</b>	<b>/req/Construction/OuterCeilingSurface</b>
A	The Implementation Specification SHALL contain an element with the same definition as the OuterCeilingSurface UML class as documented in the OuterCeilingSurface section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the OuterCeilingSurface UML class as documented in the OuterCeilingSurface section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the OuterCeilingSurface UML class; including the name, definition, type, and cardinality of those documented in the OuterCeilingSurface section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the OuterCeilingSurface UML class; including the name, definition, type, and cardinality of those documented in the OuterCeilingSurface section of the <a href="#">Construction Data Dictionary</a> .

### **Class 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»

#### **Relations**

#### **Attributes**

### **11.9.25. Class OuterFloorSurface**

<b>Requirement 145</b>	<b>/req/Construction/OuterFloorSurface</b>
------------------------	--------------------------------------------

A	The Implementation Specification SHALL contain an element with the same definition as the OuterFloorSurface UML class as documented in the OuterFloorSurface section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the OuterFloorSurface UML class as documented in the OuterFloorSurface section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the OuterFloorSurface UML class; including the name, definition, type, and cardinality of those documented in the OuterFloorSurface section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the OuterFloorSurface UML class; including the name, definition, type, and cardinality of those documented in the OuterFloorSurface section of the <a href="#">Construction Data Dictionary</a> .

#### Class 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»

#### Relations

#### Attributes

### 11.9.26. Class RoofSurface

Requirement 146	/req/Construction/RoofSurface
A	The Implementation Specification SHALL contain an element with the same definition as the RoofSurface UML class as documented in the RoofSurface section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the RoofSurface UML class as documented in the RoofSurface section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the RoofSurface UML class; including the name, definition, type, and cardinality of those documented in the RoofSurface section of the <a href="#">Construction Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the RoofSurface UML class; including the name, definition, type, and cardinality of those documented in the RoofSurface section of the <a href="#">Construction Data Dictionary</a> .
---	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class RoofSurface

Definition: A RoofSurface is a surface that delimits major roof parts of a construction.

Subclass Of: [AbstractConstructionSurface](#)

Stereotype: «FeatureType»

#### Relations

#### Attributes

### 11.9.27. Class WallSurface

Requirement 147	/req/Construction/WallSurface
A	The Implementation Specification SHALL contain an element with the same definition as the WallSurface UML class as documented in the WallSurface section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the WallSurface UML class as documented in the WallSurface section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the WallSurface UML class; including the name, definition, type, and cardinality of those documented in the WallSurface section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the WallSurface UML class; including the name, definition, type, and cardinality of those documented in the WallSurface section of the <a href="#">Construction Data Dictionary</a> .

### Class WallSurface

Definition: A WallSurface is a surface that is part of the building facade visible from the outside.

Subclass Of: [AbstractConstructionSurface](#)

Stereotype: «FeatureType»

#### Relations

#### Attributes

### 11.9.28. Class Window

Requirement 148	/req/Construction/Window
-----------------	--------------------------

A	The Implementation Specification SHALL contain an element with the same definition as the Window UML class as documented in the Window section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Window UML class as documented in the Window section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Window UML class; including the name, definition, type, and cardinality of those documented in the Window section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Window UML class; including the name, definition, type, and cardinality of those documented in the Window section of the <a href="#">Construction Data Dictionary</a> .

## Class 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»

## Relations

Relation Name: boundary

Cardinality: \*

Target Class: [WindowSurface](#)

Definition:

## Attributes

Attribute Name: class

Value Type: [WindowClassName](#)

Definition: Indicates the specific type of the Window.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: [WindowFunctionName](#)

Definition: Specifies the intended purposes of the Window.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: [WindowUsageName](#)

Definition: Specifies the actual uses of the Window.

Multiplicity: [0..\*]

Stereotype: «Property»

### 11.9.29. Class WindowClassName

Requirement 149	/req/Construction/WindowClassName
A	Any use of the WindowClassName type in an Implementation Specification SHALL be restricted to the valid values specified in the WindowClassName UML class as documented in the WindowClassName section of the <a href="#">Construction Data Dictionary</a> .

#### Class WindowClassName

Definition: WindowClassName is a code list used to further classify a Window.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.9.30. Class WindowFunctionValue

Requirement 150	/req/Construction/WindowFunctionValue
A	Any use of the WindowFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WindowFunctionValue UML class as documented in the WindowFunctionValue section of the <a href="#">Construction Data Dictionary</a> .

#### Class WindowFunctionValue

Definition: WindowFunctionValue is a code list that enumerates the different purposes of a Window.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.9.31. Class WindowSurface

Requirement 151	/req/Construction/WindowSurface
A	The Implementation Specification SHALL contain an element with the same definition as the WindowSurface UML class as documented in the WindowSurface section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the WindowSurface UML class as documented in the WindowSurface section of the <a href="#">Construction Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the WindowSurface UML class; including the name, definition, type, and cardinality of those documented in the WindowSurface section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the WindowSurface UML class; including the name, definition, type, and cardinality of those documented in the WindowSurface section of the <a href="#">Construction Data Dictionary</a> .

### Class 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»

#### Relations

#### Attributes

### 11.9.32. Class WindowUsageValue

Requirement 152	/req/Construction/WindowUsageValue
A	Any use of the WindowUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WindowUsageValue UML class as documented in the WindowUsageValue section of the <a href="#">Construction Data Dictionary</a> .

### Class WindowUsageValue

Definition: WindowUsageValue is a code list that enumerates the different uses of a Window.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.9.33. Class ConditionOfConstructionValue

Requirement 153	/req/Construction/ConditionOfConstructionValue
A	Any use of the ConditionOfConstructionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the ConditionOfConstructionValue UML class as documented in the ConditionOfConstructionValue section of the <a href="#">Construction Data Dictionary</a> .

## **Class ConditionOfConstructionValue**

Definition: ConditionOfConstructionValue enumerates different conditions of a construction.

[cf. INSPIRE]

Subclass Of: <-- section,>>

Stereotype:

### **Relations**

### **Attributes**

Attribute Name: declined

Value Type:

Definition: Indicates that the construction cannot be used under normal conditions, though its main elements (walls, roof) are still present. [cf. INSPIRE]

Multiplicity:

Stereotype:

Attribute Name: demolished

Value Type:

Definition: Indicates that the construction has been demolished. There are no more visible remains. [cf. INSPIRE]

Multiplicity:

Stereotype:

Attribute Name: functional

Value Type:

Definition: Indicates that the construction is functional. [cf. INSPIRE]

Multiplicity:

Stereotype:

Attribute Name: projected

Value Type:

Definition: Indicates that the construction is being designed. Construction works have not yet started. [cf. INSPIRE]

Multiplicity:

Stereotype:

Attribute Name: ruin

Value Type:

Definition: 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]

Multiplicity:

Stereotype:

Attribute Name: underConstruction

Value Type:

Definition: 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]

Multiplicity:

Stereotype:

### 11.9.34. Class ConstructionEvent

Requirement 154	/req/Construction/ConstructionEvent
A	Any use of the ConstructionEvent type in the Implementation Specification SHALL have the same definition as the ConstructionEvent UML class as documented in the ConstructionEvent section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ConstructionEvent UML class as documented in the ConstructionEvent section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the ConstructionEvent UML class; including the name, definition, type, and cardinality of those documented in the ConstructionEvent section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ConstructionEvent UML class; including the name, definition, type, and cardinality of those documented in the ConstructionEvent section of the <a href="#">Construction Data Dictionary</a> .

#### Class 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: <-- section,>>

Stereotype: «DataType»

#### Relations

#### Attributes

Attribute Name: dateOfEvent

Value Type: Date

Definition: Specifies the date at which the event took or will take place.

Multiplicity:

Stereotype: «Property»

Attribute Name: description

Value Type: CharacterString

Definition: Provides additional information on the event.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: event

Value Type: EventValue

Definition: Indicates the specific event type.

Multiplicity:

Stereotype: «Property»

### 11.9.35. Class Elevation

Requirement 155	/req/Construction/Elevation
A	Any use of the Elevation type in the Implementation Specification SHALL have the same definition as the Elevation UML class as documented in the Elevation section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Elevation UML class as documented in the Elevation section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Elevation UML class; including the name, definition, type, and cardinality of those documented in the Elevation section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Elevation UML class; including the name, definition, type, and cardinality of those documented in the Elevation section of the <a href="#">Construction Data Dictionary</a> .

#### Class 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: <-> section,>>

Stereotype: «DataType»

#### Relations

#### Attributes

Attribute Name: elevationReference

Value Type: ElevationReferenceValue

Definition: Specifies the level from which the elevation was measured. [cf. INSPIRE]

Multiplicity:

Stereotype: «Property»

Attribute Name: elevationValue

Value Type: DirectPosition

Definition: Specifies the value of the elevation. [cf. INSPIRE]

Multiplicity:

Stereotype: «Property»

### 11.9.36. Class Height

Requirement 156	/req/Construction/Height
-----------------	--------------------------

A	Any use of the Height type in the Implementation Specification SHALL have the same definition as the Height UML class as documented in the Height section of the <a href="#">Construction Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Height UML class as documented in the Height section of the <a href="#">Construction Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Height UML class; including the name, definition, type, and cardinality of those documented in the Height section of the <a href="#">Construction Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Height UML class; including the name, definition, type, and cardinality of those documented in the Height section of the <a href="#">Construction Data Dictionary</a> .

## Class Height

Definition: Height represents a vertical distance (measured or estimated) between a low reference and a high reference. [cf. INSPIRE]

Subclass Of: <-- section,>>

Stereotype: «DataType»

## Relations

## Attributes

Attribute Name: highReference

Value Type: ElevationReferenceValue

Definition: Indicates the high point used to calculate the value of the height. [cf. INSPIRE]

Multiplicity:

Stereotype: «Property»

Attribute Name: lowReference

Value Type: ElevationReferenceValue

Definition: Indicates the low point used to calculate the value of the height. [cf. INSPIRE]

Multiplicity:

Stereotype: «Property»

Attribute Name: status

Value Type: HeightStatusValue

Definition: Indicates the way the height has been captured. [cf. INSPIRE]

Multiplicity:

Stereotype: «Property»

Attribute Name: value

Value Type: Length

Definition: Specifies the value of the height above or below ground. [cf. INSPIRE]

Multiplicity:

Stereotype: «Property»

### 11.9.37. Class HeightStatusValue

Requirement 157	/req/Construction/HeightStatusValue
A	Any use of the HeightStatusValue type in an Implementation Specification SHALL be restricted to the valid values specified in the HeightStatusValue UML class as documented in the HeightStatusValue section of the <a href="#">Construction Data Dictionary</a> .

#### Class HeightStatusValue

Definition: HeightStatusValue enumerates the different methods used to capture a height. [cf. INSPIRE]

Subclass Of: <-- section,>>

Stereotype:

#### Relations

#### Attributes

Attribute Name: estimated

Value Type:

Definition: Indicates that the height has been estimated and not measured. [cf. INSPIRE]

Multiplicity:

Stereotype:

Attribute Name: measured

Value Type:

Definition: Indicates that the height has been (directly or indirectly) measured. [cf. INSPIRE]

Multiplicity:

Stereotype:

### 11.9.38. Class RelationToConstruction

Requirement 158	/req/Construction/RelationToConstruction
A	Any use of the RelationToConstruction type in an Implementation Specification SHALL be restricted to the valid values specified in the RelationToConstruction UML class as documented in the RelationToConstruction section of the <a href="#">Construction Data Dictionary</a> .

#### Class RelationToConstruction

Definition: RelationToConstruction is an enumeration used to describe whether an installation is positioned inside and/or outside of a construction.

Subclass Of: <-- section,>>

Stereotype:

#### Relations

## Attributes

Attribute Name:	inside
Value Type:	
Definition:	Indicates that the installation is positioned inside of the construction.
Multiplicity:	
Stereotype:	
Attribute Name:	outside
Value Type:	
Definition:	Indicates that the installation is positioned outside of the construction.
Multiplicity:	
Stereotype:	
Attribute Name:	bothInsideAndOutside
Value Type:	
Definition:	Indicates that the installation is positioned inside as well as outside of the construction.
Multiplicity:	
Stereotype:	

### 11.9.39. Additional Information

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

## 11.10. Dynamizer

### Requirements Class

<http://www.opengis.net/spec/CityGML/3.0/req/req-class-dynamizer>

Target type	Conceptual Model
Dependency	<a href="/req/req-class-core">/req/req-class-core</a>

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](#).

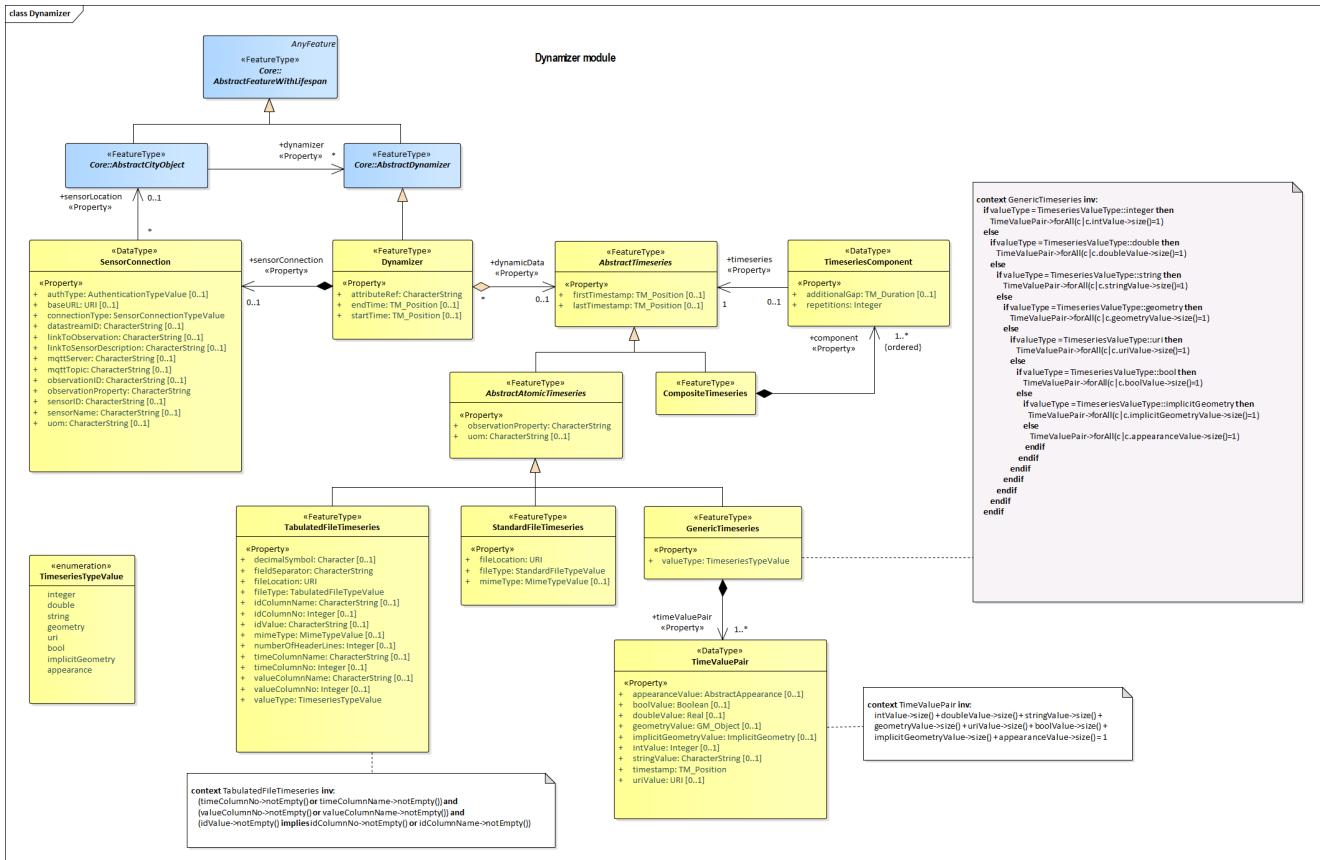


Figure 23. UML diagram of the Dynamizer Model.

### 11.10.1. Dynamizer 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.

**Stereotype:** «ApplicationSchema»

**Parent Package:** CityGML

## 11.10.2. Class AbstractAtomicTimeseries

Requirement 159	/req/Dynamizer/AbstractAtomicTimeSeries
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractAtomicTimeSeries UML class as documented in the AbstractAtomicTimeSeries section of the <a href="#">Dynamizer Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractAtomicTimeSeries UML class as documented in the AbstractAtomicTimeSeries section of the <a href="#">Dynamizer Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractAtomicTimeSeries UML class; including the name, definition, type, and cardinality of those documented in the AbstractAtomicTimeSeries section of the <a href="#">Dynamizer Data Dictionary</a> .

### Class 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»

### Relations

### Attributes

Attribute Name: observationProperty

Value Type: CharacterString

Definition: Specifies the phenomenon for which the atomic timeseries provides observation values.

Multiplicity:

Stereotype: «Property»

Attribute Name: uom

Value Type: CharacterString

Definition: Specifies the unit of measurement of the observation values.

Multiplicity: [0..1]

Stereotype: «Property»

## 11.10.3. Class AbstractTimeseries

Requirement 160	/req/Dynamizer/AbstractTimeSeries
-----------------	-----------------------------------

A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractTimeSeries UML class as documented in the AbstractTimeSeries section of the <a href="#">Dynamizer Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractTimeSeries UML class as documented in the AbstractTimeSeries section of the <a href="#">Dynamizer Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractTimeSeries UML class; including the name, definition, type, and cardinality of those documented in the AbstractTimeSeries section of the <a href="#">Dynamizer Data Dictionary</a> .

### Class AbstractTimeseries

Definition: AbstractTimeseries is the abstract superclass representing any type of timeseries data.

Subclass Of: <-- section,>>

Stereotype: «FeatureType»

### Relations

### Attributes

Attribute Name: firstTimestamp

Value Type: TM\_Position

Definition: Specifies the beginning of the timeseries.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: lastTimestamp

Value Type: TM\_Position

Definition: Specifies the end of the timeseries.

Multiplicity: [0..1]

Stereotype: «Property»

### 11.10.4. Class AuthenticationTypeValue

Requirement 161	/req/Dynamizer/AuthenticationTypeValue
A	Any use of the AuthenticationTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuthenticationTypeValue UML class as documented in the AuthenticationTypeValue section of the <a href="#">Dynamizer Data Dictionary</a> .

### **Class 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: <-- section,>>

Stereotype: «CodeList»

### **Relations**

### **Attributes**

## **11.10.5. Class CompositeTimeseries**

<b>Requirement 162</b>	<b>/req/Dynamizer/CompositeTimeseries</b>
A	The Implementation Specification SHALL contain an element with the same definition as the CompositeTimeseries UML class as documented in the CompositeTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the CompositeTimeseries UML class as documented in the CompositeTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the CompositeTimeseries UML class; including the name, definition, type, and cardinality of those documented in the CompositeTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the CompositeTimeseries UML class; including the name, definition, type, and cardinality of those documented in the CompositeTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .

### **Class 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»

### **Relations**

Relation Name: component

Cardinality: 1..\*

Target Class: [TimeseriesComponent](#)

Definition: Relates to the atomic and composite timeseries that are part of the CompositeTimeseries. The referenced timeseries are sequentially ordered.

### **Attributes**

## 11.10.6. Class Dynamizer

Requirement 163	/req/Dynamizer/Dynamizer
A	The Implementation Specification SHALL contain an element with the same definition as the Dynamizer UML class as documented in the Dynamizer section of the <a href="#">Dynamizer Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Dynamizer UML class as documented in the Dynamizer section of the <a href="#">Dynamizer Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Dynamizer UML class; including the name, definition, type, and cardinality of those documented in the Dynamizer section of the <a href="#">Dynamizer Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Dynamizer UML class; including the name, definition, type, and cardinality of those documented in the Dynamizer section of the <a href="#">Dynamizer Data Dictionary</a> .

### Class 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»

### Relations

Relation Name: dynamicData

Cardinality: 0..1

Target Class: [AbstractTimeseries](#)

Definition: Relates to the timeseries data that is given either inline within a CityGML dataset or by a link to an external file containing timeseries data.

Relation Name: sensorConnection

Cardinality: 0..1

Target Class: [SensorConnection](#)

Definition: Relates to the sensor API that delivers timeseries data.

### Attributes

Attribute Name:	attributeRef
Value Type:	CharacterString
Definition:	Specifies the attribute of a CityGML feature whose value is overridden or replaced by the (dynamic) values specified by the Dynamizer.
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	endTime
Value Type:	TM_Position
Definition:	Specifies the end of the time span for which the Dynamizer provides dynamic values.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	startTime
Value Type:	TM_Position
Definition:	Specifies the beginning of the time span for which the Dynamizer provides dynamic values.
Multiplicity:	[0..1]
Stereotype:	«Property»

## 11.10.7. Class GenericTimeseries

Requirement 164	/req/Dynamizer/GenericTimeseries
A	The Implementation Specification SHALL contain an element with the same definition as the GenericTimeseries UML class as documented in the GenericTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GenericTimeseries UML class as documented in the GenericTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the GenericTimeseries UML class; including the name, definition, type, and cardinality of those documented in the GenericTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GenericTimeseries UML class; including the name, definition, type, and cardinality of those documented in the GenericTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .

## **Class 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»

## **Relations**

Relation Name: timeValuePair

Cardinality: 1..\*

Target Class: [TimeValuePair](#)

Definition: Relates to the time-value-pairs that are part of the GenericTimeseries.

## **Attributes**

Attribute Name: valueType

Value Type: TimeseriesTypeValue

Definition: Indicates the specific type of all time-value-pairs that are part of the GenericTimeseries.

Multiplicity:

Stereotype: «Property»

## **11.10.8. Class SensorConnectionTypeValue**

Requirement 165	/req/Dynamizer/SensorConnectionTypeValue
A	Any use of the SensorConnectionTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SensorConnectionTypeValue UML class as documented in the SensorConnectionTypeValue section of the <a href="#">Dynamizer Data Dictionary</a> .

## **Class 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: <-- section,>>

Stereotype: «CodeList»

## **Relations**

## **Attributes**

## **11.10.9. Class StandardFileTimeseries**

Requirement 166	/req/Dynamizer/StandardFileTimeseries
A	The Implementation Specification SHALL contain an element with the same definition as the StandardFileTimeseries UML class as documented in the StandardFileTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the StandardFileTimeseries UML class as documented in the StandardFileTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the StandardFileTimeseries UML class; including the name, definition, type, and cardinality of those documented in the StandardFileTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the StandardFileTimeseries UML class; including the name, definition, type, and cardinality of those documented in the StandardFileTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .

### Class 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»

### Relations

### Attributes

Attribute Name: fileLocation

Value Type: URI

Definition: Specifies the URI that points to the external timeseries file.

Multiplicity:

Stereotype: «Property»

Attribute Name: fileType

Value Type: StandardFileTypeValue

Definition: Specifies the format used to represent the timeseries data.

Multiplicity:

Stereotype: «Property»

Attribute Name: mimeType

Value Type: MimeTypeValue

Definition: Specifies the MIME type of the external timeseries file.

Multiplicity: [0..1]

Stereotype: «Property»

### 11.10.10. Class StandardFileTypeValue

Requirement 167	/req/Dynamizer/StandardFileTypeValue
-----------------	--------------------------------------

A	Any use of the StandardFileTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the StandardFileTypeValue UML class as documented in the StandardFileTypeValue section of the <a href="#">Dynamizer Data Dictionary</a> .
---	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class 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: <-->

StereoType: «CodeList»

#### Relations

#### Attributes

### 11.10.11. Class TabulatedFileTimeseries

Requirement 168	/req/Dynamizer/TabulatedFileTimeseries
A	The Implementation Specification SHALL contain an element with the same definition as the TabulatedFileTimeseries UML class as documented in the TabulatedFileTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TabulatedFileTimeseries UML class as documented in the TabulatedFileTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TabulatedFileTimeseries UML class; including the name, definition, type, and cardinality of those documented in the TabulatedFileTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TabulatedFileTimeseries UML class; including the name, definition, type, and cardinality of those documented in the TabulatedFileTimeseries section of the <a href="#">Dynamizer Data Dictionary</a> .

## Class 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»

## Relations

## Attributes

Attribute Name: decimalSymbol

Value Type: Character

Definition: Indicates which symbol is used to separate the integer part from the fractional part of a decimal number.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: fieldSeparator

Value Type: CharacterString

Definition: Indicates which symbol is used to separate the individual values in the tabulated file.

Multiplicity:

Stereotype: «Property»

Attribute Name: fileLocation

Value Type: URI

Definition: Specifies the URI that points to the external timeseries file.

Multiplicity:

Stereotype: «Property»

Attribute Name: fileType

Value Type: TabulatedFileTypeValue

Definition: Specifies the format used to represent the timeseries data.

Multiplicity:

Stereotype: «Property»

Attribute Name: idColumnName

Value Type: CharacterString

Definition: Specifies the name of the column that stores the identifier of the time-value-pair.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	idColumnNo
Value Type:	Integer
Definition:	Specifies the number of the column that stores the identifier of the time-value-pair.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	idValue
Value Type:	CharacterString
Definition:	Specifies the value of the identifier for which the time-value-pairs are to be selected.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	mimeType
Value Type:	MimeTypeValue
Definition:	Specifies the MIME type of the external timeseries file.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	numberOfHeaderLines
Value Type:	Integer
Definition:	Indicates the number of lines at the beginning of the tabulated file that represent headers.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	timeColumnName
Value Type:	CharacterString
Definition:	Specifies the name of the column that stores the timestamp of the time-value-pair.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	timeColumnNo
Value Type:	Integer
Definition:	Specifies the number of the column that stores the timestamp of the time-value-pair.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	valueColumnName
Value Type:	CharacterString
Definition:	Specifies the name of the column that stores the value of the time-value-pair.
Multiplicity:	[0..1]
Stereotype:	«Property»

Attribute Name:	valueColumnNo
Value Type:	Integer
Definition:	Specifies the number of the column that stores the value of the time-value-pair.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	valueType
Value Type:	TimeseriesTypeValue
Definition:	Indicates the specific type of the timeseries data.
Multiplicity:	
Stereotype:	«Property»

### 11.10.12. Class TabulatedFileTypeValue

Requirement 169	/req/Dynamizer/TabulatedFileTypeValue
A	Any use of the TabulatedFileTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TabulatedFileTypeValue UML class as documented in the TabulatedFileTypeValue section of the <a href="#">Dynamizer Data Dictionary</a> .

#### Class TabulatedFileTypeValue

Definition: TabulatedFileTypeValue is a code list used to specify the data format of the referenced external tabulated data file.

Subclass Of: <-- section,>>

StereoType: «CodeList»

#### Relations

#### Attributes

### 11.10.13. Class SensorConnection

Requirement 170	/req/Dynamizer/SensorConnection
A	Any use of the SensorConnection type in the Implementation Specification SHALL have the same definition as the SensorConnection UML class as documented in the SensorConnection section of the <a href="#">Dynamizer Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the SensorConnection UML class as documented in the SensorConnection section of the <a href="#">Dynamizer Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the SensorConnection UML class; including the name, definition, type, and cardinality of those documented in the SensorConnection section of the <a href="#">Dynamizer Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the SensorConnection UML class; including the name, definition, type, and cardinality of those documented in the SensorConnection section of the <a href="#">Dynamizer Data Dictionary</a> .
---	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Class 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: <-->

Stereotype: «DataType»

### Relations

Relation Name: sensorLocation

Cardinality: 0..1

Target Class: [AbstractCityObject](#)

Definition: Relates the sensor to the city object where it is located.

### Attributes

Attribute Name: authType

Value Type: AuthenticationTypeValue

Definition: Specifies the type of authentication required to be able to access the Sensor API.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: baseURL

Value Type: URI

Definition: Specifies the base URL of the Sensor API request.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: connectionType

Value Type: SensorConnectionTypeValue

Definition: Indicates the type of Sensor API to which the SensorConnection refers.

Multiplicity:

Stereotype: «Property»

Attribute Name: datastreamID

Value Type: CharacterString

Definition: Specifies the datastream that is retrieved by the SensorConnection.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	linkToObservation
Value Type:	CharacterString
Definition:	Specifies the complete URL to the observation request.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	linkToSensorDescription
Value Type:	CharacterString
Definition:	Specifies the complete URL to the sensor description request.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	mqttServer
Value Type:	CharacterString
Definition:	Specifies the name of the MQTT Server. This attribute is relevant when the MQTT Protocol is used to connect to a Sensor API.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	mqttTopic
Value Type:	CharacterString
Definition:	Names the specific datastream that is retrieved by the SensorConnection.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	observationID
Value Type:	CharacterString
Definition:	Specifies the unique identifier of the observation that is retrieved by the SensorConnection.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	observationProperty
Value Type:	CharacterString
Definition:	Specifies the phenomenon for which the SensorConnection provides observations.
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	sensorID
Value Type:	CharacterString
Definition:	Specifies the unique identifier of the sensor from which the SensorConnection retrieves observations.
Multiplicity:	[0..1]
Stereotype:	«Property»

Attribute Name:	sensorName
Value Type:	CharacterString
Definition:	Specifies the name of the sensor from which the SensorConnection retrieves observations.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	uom
Value Type:	CharacterString
Definition:	Specifies the unit of measurement of the observations.
Multiplicity:	[0..1]
Stereotype:	«Property»

### 11.10.14. Class TimeseriesComponent

Requirement 171	/req/Dynamizer/TimeseriesComponent
A	Any use of the TimeseriesComponent type in the Implementation Specification SHALL have the same definition as the TimeseriesComponent UML class as documented in the TimeseriesComponent section of the <a href="#">Dynamizer Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TimeseriesComponent UML class as documented in the TimeseriesComponent section of the <a href="#">Dynamizer Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TimeseriesComponent UML class; including the name, definition, type, and cardinality of those documented in the TimeseriesComponent section of the <a href="#">Dynamizer Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TimeseriesComponent UML class; including the name, definition, type, and cardinality of those documented in the TimeseriesComponent section of the <a href="#">Dynamizer Data Dictionary</a> .

#### Class TimeseriesComponent

Definition: TimeseriesComponent represents an element of a CompositeTimeseries.

Subclass Of: <-- section,>>

Stereotype: «DataType»

#### Relations

Relation Name: timeseries

Cardinality: 1

Target Class: [AbstractTimeseries](#)

Definition: Relates the TimeseriesComponent to a timeseries.

#### Attributes

Attribute Name:	additionalGap
Value Type:	TM_Duration
Definition:	Specifies how much extra time is added after all repetitions as an additional gap.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	repetitions
Value Type:	Integer
Definition:	Specifies how often the timeseries that is referenced by the TimeseriesComponent should be iterated.
Multiplicity:	
Stereotype:	«Property»

## 11.10.15. Class TimeseriesTypeValue

Requirement 172	/req/Dynamizer/TimeseriesTypeValue
A	Any use of the TimeseriesTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TimeseriesTypeValue UML class as documented in the TimeseriesTypeValue section of the <a href="#">Dynamizer Data Dictionary</a> .

### Class TimeseriesTypeValue

Definition: TimeseriesTypeValue enumerates the possible value types for GenericTimeseries and TimeValuePair.

Subclass Of: <-- section,>>

Stereotype:

### Relations

### Attributes

Attribute Name: integer

Value Type:

Definition: Indicates that the values of the GenericTimeseries are of type "Integer".

Multiplicity:

Stereotype:

Attribute Name: double

Value Type:

Definition: Indicates that the values of the GenericTimeseries are of type "Double".

Multiplicity:

Stereotype:

Attribute Name: string

Value Type:

Definition: Indicates that the values of the GenericTimeseries are of type "String".

Multiplicity:

Stereotype:

Attribute Name:	geometry
Value Type:	
Definition:	Indicates that the values of the GenericTimeseries are geometries.
Multiplicity:	
Stereotype:	
Attribute Name:	uri
Value Type:	
Definition:	Indicates that the values of the GenericTimeseries are of type "URI".
Multiplicity:	
Stereotype:	
Attribute Name:	bool
Value Type:	
Definition:	Indicates that the values of the GenericTimeseries are of type "Boolean".
Multiplicity:	
Stereotype:	
Attribute Name:	implicitGeometry
Value Type:	
Definition:	Indicates that the values of the GenericTimeseries are of type "ImplicitGeometry".
Multiplicity:	
Stereotype:	
Attribute Name:	appearance
Value Type:	
Definition:	Indicates that the values of the GenericTimeseries are of type "Appearance".
Multiplicity:	
Stereotype:	

### 11.10.16. Class TimeValuePair

Requirement 173	/req/Dynamizer/TimeValuePair
A	Any use of the TimeValuePair type in the Implementation Specification SHALL have the same definition as the TimeValuePair UML class as documented in the TimeValuePair section of the <a href="#">Dynamizer Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TimeValuePair UML class as documented in the TimeValuePair section of the <a href="#">Dynamizer Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TimeValuePair UML class; including the name, definition, type, and cardinality of those documented in the TimeValuePair section of the <a href="#">Dynamizer Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the TimeValuePair UML class; including the name, definition, type, and cardinality of those documented in the TimeValuePair section of the <a href="#">Dynamizer Data Dictionary</a> .
---	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Class 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: <-->

Stereotype: «DataType»

## Relations

### Attributes

Attribute Name: appearanceValue

Value Type: AbstractAppearance

Definition: Specifies the "Appearance" value of the TimeValuePair.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: boolValue

Value Type: Boolean

Definition: Specifies the "Boolean" value of the TimeValuePair.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: doubleValue

Value Type: Real

Definition: Specifies the "Double" value of the TimeValuePair.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: geometryValue

Value Type: GM\_Object

Definition: Specifies the geometry value of the TimeValuePair.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: implicitGeometryValue

Value Type: ImplicitGeometry

Definition: Specifies the "ImplicitGeometry" value of the TimeValuePair.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: intValue

Value Type: Integer

Definition: Specifies the "Integer" value of the TimeValuePair.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	stringValue
Value Type:	CharacterString
Definition:	Specifies the "String" value of the TimeValuePair.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	timestamp
Value Type:	TM_Position
Definition:	Specifies the timepoint at which the value of the TimeValuePair is valid.
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	uriValue
Value Type:	URI
Definition:	Specifies the "URI" value of the TimeValuePair.
Multiplicity:	[0..1]
Stereotype:	«Property»

### 11.10.17. Additional Information

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

## 11.11. Generics

Requirements Class	
<a href="http://www.opengis.net/spec/CityGML/3.0/req/req-class-generics">http://www.opengis.net/spec/CityGML/3.0/req/req-class-generics</a>	
Target type	Conceptual Model
Dependency	<a href="/req/req-class-core">/req/req-class-core</a>

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](#).

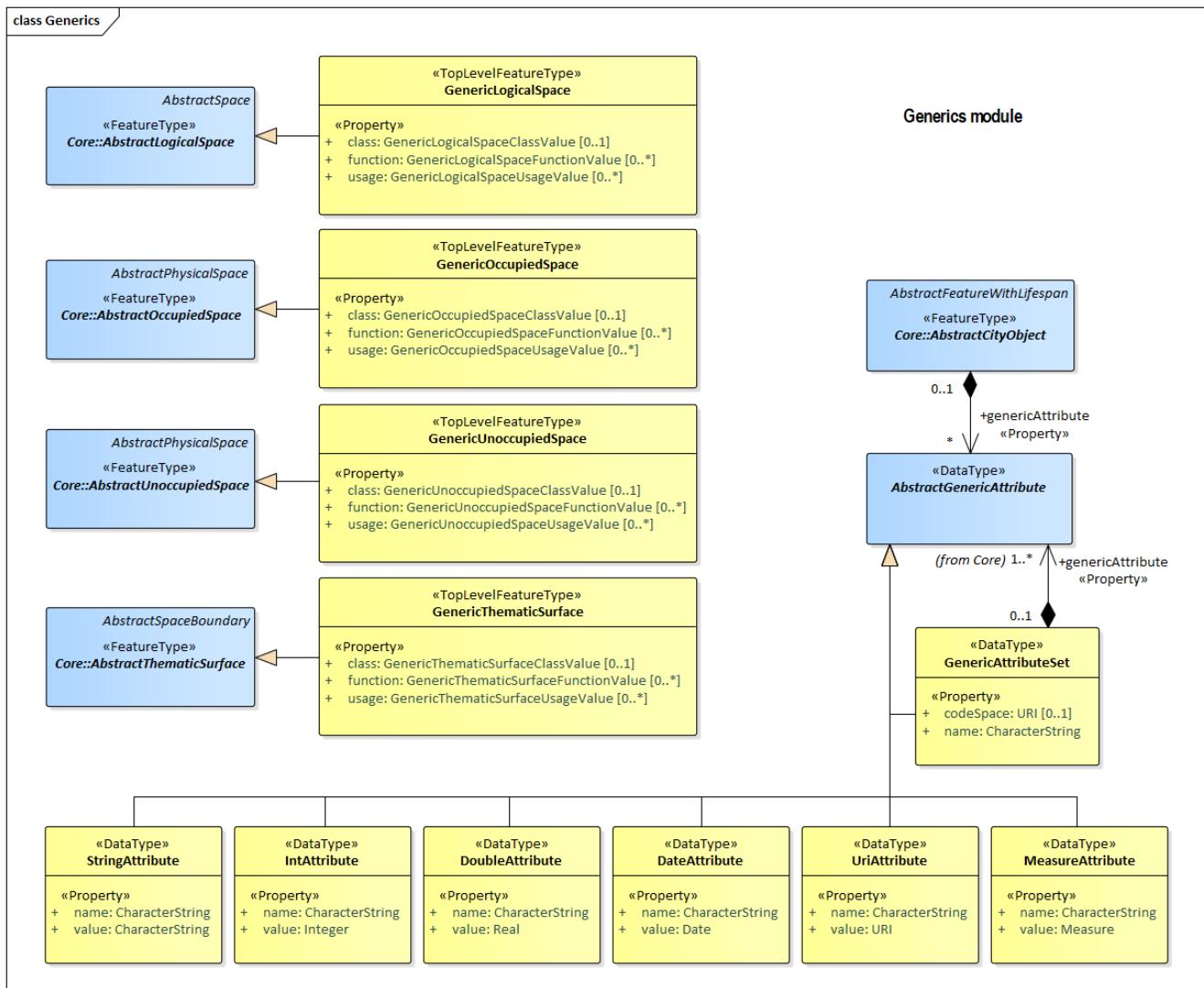


Figure 24. UML diagram of the Generics Model.

### 11.11.1. Generics 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.

**Stereotype:** `«ApplicationSchema»`

**Parent Package:** CityGML

### 11.11.2. Class GenericLogicalSpace

Requirement 174	/req/Generics/GenericLogicalSpace
-----------------	-----------------------------------

A	The Implementation Specification SHALL contain an element with the same definition as the GenericLogicalSpace UML class as documented in the GenericLogicalSpace section of the <a href="#">Generics Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GenericLogicalSpace UML class as documented in the GenericLogicalSpace section of the <a href="#">Generics Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the GenericLogicalSpace UML class; including the name, definition, type, and cardinality of those documented in the GenericLogicalSpace section of the <a href="#">Generics Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GenericLogicalSpace UML class; including the name, definition, type, and cardinality of those documented in the GenericLogicalSpace section of the <a href="#">Generics Data Dictionary</a> .

## Class GenericLogicalSpace

Definition: A GenericLogicalSpace is a space that is not represented by any explicitly modelled AbstractLogicalSpace subclass within CityGML.

Subclass Of: [AbstractLogicalSpace](#)

Stereotype: «TopLevelFeatureType»

## Relations

## Attributes

Attribute Name: class

Value Type: GenericLogicalSpaceClassValue

Definition: Indicates the specific type of the GenericLogicalSpace.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: GenericLogicalSpaceFunctionValue

Definition: Specifies the intended purposes of the GenericLogicalSpace.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: GenericLogicalSpaceUsageValue

Definition: Specifies the actual uses of the GenericLogicalSpace.

Multiplicity: [0..\*]

Stereotype: «Property»

### 11.11.3. Class GenericLogicalSpaceClassValue

Requirement 175	/req/Generics/GenericLogicalSpaceClassValue
A	Any use of the GenericLogicalSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericLogicalSpaceClassValue UML class as documented in the GenericLogicalSpaceClassValue section of the <a href="#">Generics Data Dictionary</a> .

#### Class GenericLogicalSpaceClassValue

Definition: GenericLogicalSpaceClassValue is a code list used to further classify a GenericLogicalSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.11.4. Class GenericLogicalSpaceFunctionValue

Requirement 176	/req/Generics/GenericLogicalSpaceFunctionValue
A	Any use of the GenericLogicalSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericLogicalSpaceFunctionValue UML class as documented in the GenericLogicalSpaceFunctionValue section of the <a href="#">Generics Data Dictionary</a> .

#### Class GenericLogicalSpaceFunctionValue

Definition: GenericLogicalSpaceFunctionValue is a code list that enumerates the different purposes of a GenericLogicalSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.11.5. Class GenericLogicalSpaceUsageValue

Requirement 177	/req/Generics/GenericLogicalSpaceUsageValue
A	Any use of the GenericLogicalSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericLogicalSpaceUsageValue UML class as documented in the GenericLogicalSpaceUsageValue section of the <a href="#">Generics Data Dictionary</a> .

### **Class GenericLogicalSpaceUsageValue**

Definition: GenericLogicalSpaceUsageValue is a code list that enumerates the different uses of a GenericLogicalSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

## **11.11.6. Class GenericOccupiedSpace**

<b>Requirement 178</b>	<b>/req/Generics/GenericOccupiedSpace</b>
A	The Implementation Specification SHALL contain an element with the same definition as the GenericOccupiedSpace UML class as documented in the GenericOccupiedSpace section of the <a href="#">Generics Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GenericOccupiedSpace UML class as documented in the GenericOccupiedSpace section of the <a href="#">Generics Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the GenericOccupiedSpace UML class; including the name, definition, type, and cardinality of those documented in the GenericOccupiedSpace section of the <a href="#">Generics Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GenericOccupiedSpace UML class; including the name, definition, type, and cardinality of those documented in the GenericOccupiedSpace section of the <a href="#">Generics Data Dictionary</a> .

### **Class GenericOccupiedSpace**

Definition: A GenericOccupiedSpace is a space that is not represented by any explicitly modelled AbstractOccupiedSpace subclass within CityGML.

Subclass Of: [AbstractOccupiedSpace](#)

Stereotype: «TopLevelFeatureType»

#### **Relations**

#### **Attributes**

Attribute Name: class

Value Type: GenericOccupiedSpaceClassValue

Definition: Indicates the specific type of the GenericOccupiedSpace.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	function
Value Type:	GenericOccupiedSpaceFunctionValue
Definition:	Specifies the intended purposes of the GenericOccupiedSpace.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	GenericOccupiedSpaceUsageValue
Definition:	Specifies the actual uses of the GenericOccupiedSpace.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.11.7. Class GenericOccupiedSpaceClassValue

Requirement 179	/req/Generics/GenericOccupiedSpaceClassValue
A	Any use of the GenericOccupiedSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericOccupiedSpaceClassValue UML class as documented in the GenericOccupiedSpaceClassValue section of the <a href="#">Generics Data Dictionary</a> .

#### Class GenericOccupiedSpaceClassValue

Definition: GenericOccupiedSpaceClassValue is a code list used to further classify a GenericOccupiedSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.11.8. Class GenericOccupiedSpaceFunctionValue

Requirement 180	/req/Generics/GenericOccupiedSpaceFunctionValue
A	Any use of the GenericOccupiedSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericOccupiedSpaceFunctionValue UML class as documented in the GenericOccupiedSpaceFunctionValue section of the <a href="#">Generics Data Dictionary</a> .

#### Class GenericOccupiedSpaceFunctionValue

Definition: GenericOccupiedSpaceFunctionValue is a code list that enumerates the different purposes of a GenericOccupiedSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

## Attributes

### 11.11.9. Class GenericOccupiedSpaceUsageValue

Requirement 181	/req/Generics/GenericOccupiedSpaceUsageValue
A	Any use of the GenericOccupiedSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericOccupiedSpaceUsageValue UML class as documented in the GenericOccupiedSpaceUsageValue section of the <a href="#">Generics Data Dictionary</a> .

#### Class GenericOccupiedSpaceUsageValue

Definition: GenericOccupiedSpaceUsageValue is a code list that enumerates the different uses of a GenericOccupiedSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.11.10. Class GenericThematicSurface

Requirement 182	/req/Generics/GenericThematicSurface
A	The Implementation Specification SHALL contain an element with the same definition as the GenericThematicSurface UML class as documented in the GenericThematicSurface section of the <a href="#">Generics Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GenericThematicSurface UML class as documented in the GenericThematicSurface section of the <a href="#">Generics Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the GenericThematicSurface UML class; including the name, definition, type, and cardinality of those documented in the GenericThematicSurface section of the <a href="#">Generics Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GenericThematicSurface UML class; including the name, definition, type, and cardinality of those documented in the GenericThematicSurface section of the <a href="#">Generics Data Dictionary</a> .

### **Class GenericThematicSurface**

Definition: A GenericThematicSurface is a surface that is not represented by any explicitly modelled AbstractThematicSurface subclass within CityGML.

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «TopLevelFeatureType»

### **Relations**

### **Attributes**

Attribute Name: class

Value Type: GenericThematicSurfaceClassName

Definition: Indicates the specific type of the GenericThematicSurface.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: GenericThematicSurfaceFunctionName

Definition: Specifies the intended purposes of the GenericThematicSurface.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: GenericThematicSurfaceUsageName

Definition: Specifies the actual uses of the GenericThematicSurface.

Multiplicity: [0..\*]

Stereotype: «Property»

## **11.11.11. Class GenericThematicSurfaceClassName**

Requirement 183	/req/Generics/GenericThematicSurfaceClassName
A	Any use of the GenericThematicSurfaceClassName type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericThematicSurfaceClassName UML class as documented in the GenericThematicSurfaceClassName section of the <a href="#">Generics Data Dictionary</a> .

### **Class GenericThematicSurfaceClassName**

Definition: GenericThematicSurfaceClassName is a code list used to further classify a GenericThematicSurface.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### **Relations**

### **Attributes**

## **11.11.12. Class GenericThematicSurfaceFunctionName**

Requirement 184	/req/Generics/GenericThematicSurfaceFunctionName
-----------------	--------------------------------------------------

A	Any use of the GenericThematicSurfaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericThematicSurfaceFunctionValue UML class as documented in the GenericThematicSurfaceFunctionValue section of the <a href="#">Generics Data Dictionary</a> .
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class GenericThematicSurfaceFunctionValue

Definition: GenericThematicSurfaceFunctionValue is a code list that enumerates the different purposes of a GenericThematicSurface.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.11.13. Class GenericThematicSurfaceUsageValue

Requirement 185	/req/Generics/GenericThematicSurfaceUsageValue
A	Any use of the GenericThematicSurfaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericThematicSurfaceUsageValue UML class as documented in the GenericThematicSurfaceUsageValue section of the <a href="#">Generics Data Dictionary</a> .

### Class GenericThematicSurfaceUsageValue

Definition: GenericThematicSurfaceUsageValue is a code list that enumerates the different uses of a GenericThematicSurface.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.11.14. Class GenericUnoccupiedSpace

Requirement 186	/req/Generics/GenericUnoccupiedSpace
A	The Implementation Specification SHALL contain an element with the same definition as the GenericUnoccupiedSpace UML class as documented in the GenericUnoccupiedSpace section of the <a href="#">Generics Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GenericUnoccupiedSpace UML class as documented in the GenericUnoccupiedSpace section of the <a href="#">Generics Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the GenericUnoccupiedSpace UML class; including the name, definition, type, and cardinality of those documented in the GenericUnoccupiedSpace section of the <a href="#">Generics Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GenericUnoccupiedSpace UML class; including the name, definition, type, and cardinality of those documented in the GenericUnoccupiedSpace section of the <a href="#">Generics Data Dictionary</a> .

### Class GenericUnoccupiedSpace

Definition: A GenericUnoccupiedSpace is a space that is not represented by any explicitly modelled AbstractUnoccupiedSpace subclass within CityGML.

Subclass Of: [AbstractUnoccupiedSpace](#)

Stereotype: «TopLevelFeatureType»

### Relations

### Attributes

Attribute Name: class

Value Type: GenericUnoccupiedSpaceClassValue

Definition: Indicates the specific type of the GenericUnoccupiedSpace.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: GenericUnoccupiedSpaceFunctionValue

Definition: Specifies the intended purposes of the GenericUnoccupiedSpace.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: GenericUnoccupiedSpaceUsageValue

Definition: Specifies the actual uses of the GenericUnoccupiedSpace.

Multiplicity: [0..\*]

Stereotype: «Property»

### 11.11.15. Class GenericUnoccupiedSpaceClassValue

Requirement 187	/req/Generics/GenericUnoccupiedSpaceClassValue
A	Any use of the GenericUnoccupiedSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericUnoccupiedSpaceClassValue UML class as documented in the GenericUnoccupiedSpaceClassValue section of the <a href="#">Generics Data Dictionary</a> .

**Class GenericUnoccupiedSpaceClassValue**

Definition: GenericUnoccupiedSpaceClassValue is a code list used to further classify a GenericUnoccupiedSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

**Relations****Attributes****11.11.16. Class GenericUnoccupiedSpaceFunctionValue**

Requirement 188	/req/Generics/GenericUnoccupiedSpaceFunctionValue
A	Any use of the GenericUnoccupiedSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericUnoccupiedSpaceFunctionValue UML class as documented in the GenericUnoccupiedSpaceFunctionValue section of the <a href="#">Generics Data Dictionary</a> .

**Class GenericUnoccupiedSpaceFunctionValue**

Definition: GenericUnoccupiedSpaceFunctionValue is a code list that enumerates the different purposes of a GenericUnoccupiedSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

**Relations****Attributes****11.11.17. Class GenericUnoccupiedSpaceUsageValue**

Requirement 189	/req/Generics/GenericUnoccupiedSpaceUsageValue
A	Any use of the GenericUnoccupiedSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericUnoccupiedSpaceUsageValue UML class as documented in the GenericUnoccupiedSpaceUsageValue section of the <a href="#">Generics Data Dictionary</a> .

**Class GenericUnoccupiedSpaceUsageValue**

Definition: GenericUnoccupiedSpaceUsageValue is a code list that enumerates the different uses of a GenericUnoccupiedSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

**Relations****Attributes**

## 11.11.18. Class DateAttribute

Requirement 190	/req/Generics/DateAttribute
A	Any use of the DateAttribute type in the Implementation Specification SHALL have the same definition as the DateAttribute UML class as documented in the DateAttribute section of the <a href="#">Generics Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the DateAttribute UML class as documented in the DateAttribute section of the <a href="#">Generics Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the DateAttribute UML class; including the name, definition, type, and cardinality of those documented in the DateAttribute section of the <a href="#">Generics Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the DateAttribute UML class; including the name, definition, type, and cardinality of those documented in the DateAttribute section of the <a href="#">Generics Data Dictionary</a> .

### Class DateAttribute

Definition: DateAttribute is a data type used to define generic attributes of type "Date".

Subclass Of: <-->

Stereotype: «DataType»

### Relations

#### Attributes

Attribute Name: name

Value Type: CharacterString

Definition: Specifies the name of the DateAttribute.

Multiplicity:

Stereotype: «Property»

Attribute Name: value

Value Type: Date

Definition: Specifies the "Date" value.

Multiplicity:

Stereotype: «Property»

## 11.11.19. Class DoubleAttribute

Requirement 191	/req/Generics/DoubleAttribute
-----------------	-------------------------------

A	Any use of the DoubleAttribute type in the Implementation Specification SHALL have the same definition as the DoubleAttribute UML class as documented in the DoubleAttribute section of the <a href="#">Generics Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the DoubleAttribute UML class as documented in the DoubleAttribute section of the <a href="#">Generics Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the DoubleAttribute UML class; including the name, definition, type, and cardinality of those documented in the DoubleAttribute section of the <a href="#">Generics Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the DoubleAttribute UML class; including the name, definition, type, and cardinality of those documented in the DoubleAttribute section of the <a href="#">Generics Data Dictionary</a> .

### Class DoubleAttribute

Definition: DoubleAttribute is a data type used to define generic attributes of type "Double".

Subclass Of: <-- section,>>

Stereotype: «DataType»

### Relations

### Attributes

Attribute Name: name

Value Type: CharacterString

Definition: Specifies the name of the DoubleAttribute.

Multiplicity:

Stereotype: «Property»

Attribute Name: value

Value Type: Real

Definition: Specifies the "Double" value.

Multiplicity:

Stereotype: «Property»

## 11.11.20. Class GenericAttributeSet

Requirement 192	/req/Generics/GenericAttributeSet
A	Any use of the GenericAttributeSet type in the Implementation Specification SHALL have the same definition as the GenericAttributeSet UML class as documented in the GenericAttributeSet section of the <a href="#">Generics Data Dictionary</a> .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GenericAttributeSet UML class as documented in the GenericAttributeSet section of the <a href="#">Generics Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the GenericAttributeSet UML class; including the name, definition, type, and cardinality of those documented in the GenericAttributeSet section of the <a href="#">Generics Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GenericAttributeSet UML class; including the name, definition, type, and cardinality of those documented in the GenericAttributeSet section of the <a href="#">Generics Data Dictionary</a> .

### Class GenericAttributeSet

Definition: A GenericAttributeSet is a named collection of generic attributes.

Subclass Of: <-- section,>>

Stereotype: «DataType»

#### Relations

Relation Name: genericAttribute

Cardinality: 1..\*

Target Class: [AbstractGenericAttribute](#)

Definition: Relates to the generic attributes that are part of the GenericAttributeSet.

#### Attributes

Attribute Name: codeSpace

Value Type: URI

Definition: Associates the GenericAttributeSet with an authority that maintains the collection of generic attributes.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: name

Value Type: CharacterString

Definition: Specifies the name of the GenericAttributeSet.

Multiplicity:

Stereotype: «Property»

### 11.11.21. Class IntAttribute

Requirement 193	/req/Generics/IntAttribute
A	Any use of the IntAttribute type in the Implementation Specification SHALL have the same definition as the IntAttribute UML class as documented in the IntAttribute section of the <a href="#">Generics Data Dictionary</a> .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the IntAttribute UML class as documented in the IntAttribute section of the <a href="#">Generics Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the IntAttribute UML class; including the name, definition, type, and cardinality of those documented in the IntAttribute section of the <a href="#">Generics Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the IntAttribute UML class; including the name, definition, type, and cardinality of those documented in the IntAttribute section of the <a href="#">Generics Data Dictionary</a> .

### Class IntAttribute

Definition: IntAttribute is a data type used to define generic attributes of type "Integer".

Subclass Of: <-- section,>>

Stereotype: «DataType»

### Relations

### Attributes

Attribute Name: name

Value Type: CharacterString

Definition: Specifies the name of the IntAttribute.

Multiplicity:

Stereotype: «Property»

Attribute Name: value

Value Type: Integer

Definition: Specifies the "Integer" value.

Multiplicity:

Stereotype: «Property»

## 11.11.22. Class MeasureAttribute

Requirement 194	/req/Generics/MeasureAttribute
A	Any use of the MeasureAttribute type in the Implementation Specification SHALL have the same definition as the MeasureAttribute UML class as documented in the MeasureAttribute section of the <a href="#">Generics Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the MeasureAttribute UML class as documented in the MeasureAttribute section of the <a href="#">Generics Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the MeasureAttribute UML class; including the name, definition, type, and cardinality of those documented in the MeasureAttribute section of the <a href="#">Generics Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the MeasureAttribute UML class; including the name, definition, type, and cardinality of those documented in the MeasureAttribute section of the <a href="#">Generics Data Dictionary</a> .

### Class MeasureAttribute

Definition: MeasureAttribute is a data type used to define generic attributes of type "Measure".

Subclass Of: <-- section,>>

Stereotype: «DataType»

### Relations

### Attributes

Attribute Name: name

Value Type: CharacterString

Definition: Specifies the name of the MeasureAttribute.

Multiplicity:

Stereotype: «Property»

Attribute Name: value

Value Type: Measure

Definition: Specifies the value of the MeasureAttribute. The value is of type "Measure", which can additionally provide the units of measure. [cf. ISO 19103]

Multiplicity:

Stereotype: «Property»

### 11.11.23. Class StringAttribute

Requirement 195	/req/Generics/StringAttribute
A	Any use of the StringAttribute type in the Implementation Specification SHALL have the same definition as the StringAttribute UML class as documented in the StringAttribute section of the <a href="#">Generics Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the StringAttribute UML class as documented in the StringAttribute section of the <a href="#">Generics Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the StringAttribute UML class; including the name, definition, type, and cardinality of those documented in the StringAttribute section of the <a href="#">Generics Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the StringAttribute UML class; including the name, definition, type, and cardinality of those documented in the StringAttribute section of the <a href="#">Generics Data Dictionary</a> .
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class StringAttribute

Definition: StringAttribute is a data type used to define generic attributes of type "String".

Subclass Of: <-- section,>>

Stereotype: «DataType»

#### Relations

#### Attributes

Attribute Name: name

Value Type: CharacterString

Definition: Specifies the name of the StringAttribute.

Multiplicity:

Stereotype: «Property»

Attribute Name: value

Value Type: CharacterString

Definition: Specifies the "String" value.

Multiplicity:

Stereotype: «Property»

### 11.11.24. Class UriAttribute

Requirement 196	/req/Generics/UriAttribute
A	Any use of the UriAttribute type in the Implementation Specification SHALL have the same definition as the UriAttribute UML class as documented in the UriAttribute section of the <a href="#">Generics Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the UriAttribute UML class as documented in the UriAttribute section of the <a href="#">Generics Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the UriAttribute UML class; including the name, definition, type, and cardinality of those documented in the UriAttribute section of the <a href="#">Generics Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the UriAttribute UML class; including the name, definition, type, and cardinality of those documented in the UriAttribute section of the <a href="#">Generics Data Dictionary</a> .

## Class UriAttribute

Definition: UriAttribute is a data type used to define generic attributes of type "URI".

Subclass Of: <-- section,>>

Stereotype: «DataType»

### Relations

#### Attributes

Attribute Name: name

Value Type: CharacterString

Definition: Specifies the name of the UriAttribute.

Multiplicity:

Stereotype: «Property»

Attribute Name: value

Value Type: URI

Definition: Specifies the "URI" value.

Multiplicity:

Stereotype: «Property»

## 11.11.25. Additional Information

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

## 11.12. Land Use

### Requirements Class

<http://www.opengis.net/spec/CityGML/3.0/req/req-class-landuse>

Target type	Conceptual Model
Dependency	<a href="/req/req-class-core">/req/req-class-core</a>

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](#).

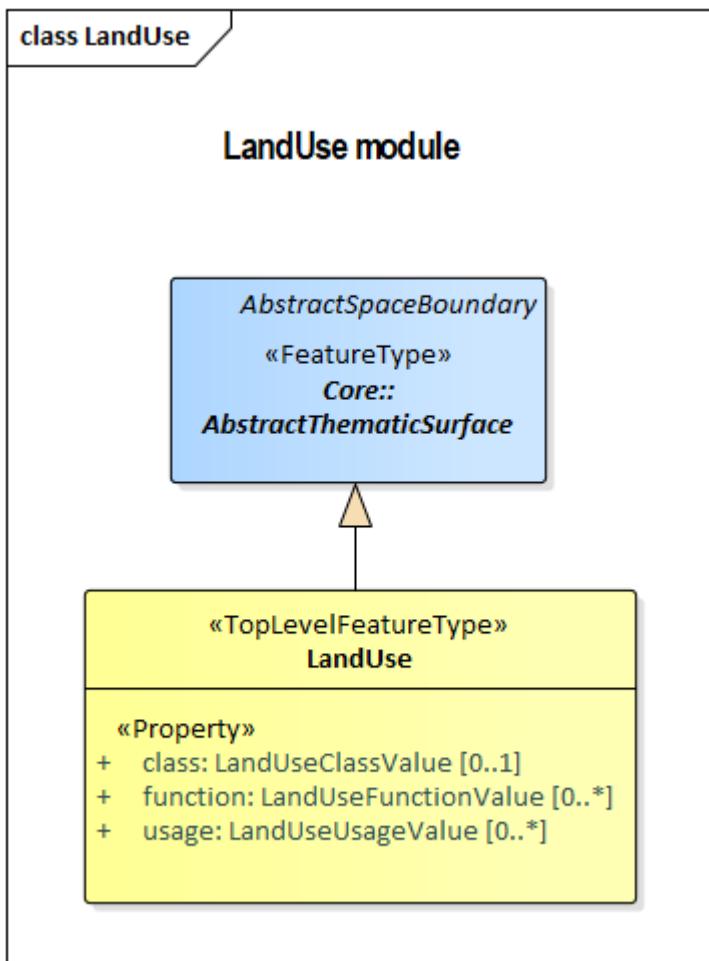


Figure 25. UML diagram of the Land Use Model.

### 11.12.1. LandUse Package

#### Package LandUse

Description: The LandUse module supports representation of areas of the earth's surface dedicated to a specific land use.

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.12.2. Class LandUse

Requirement 197	/req/LandUse/LandUse
A	The Implementation Specification SHALL contain an element with the same definition as the LandUse UML class as documented in the LandUse section of the <a href="#">LandUse Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the LandUse UML class as documented in the LandUse section of the <a href="#">LandUse Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the LandUse UML class; including the name, definition, type, and cardinality of those documented in the LandUse section of the <a href="#">LandUse Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the LandUse UML class; including the name, definition, type, and cardinality of those documented in the LandUse section of the <a href="#">LandUse Data Dictionary</a> .

### Class 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»

### Relations

### Attributes

Attribute Name: class

Value Type: LandUseClassValue

Definition: Indicates the specific type of the LandUse.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: LandUseFunctionValue

Definition: Specifies the intended purposes of the LandUse.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: LandUseUsageValue

Definition: Specifies the actual uses of the LandUse.

Multiplicity: [0..\*]

Stereotype: «Property»

### 11.12.3. Class LandUseClassValue

Requirement 198	/req/LandUse/LandUseClassValue
A	Any use of the LandUseClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the LandUseClassValue UML class as documented in the <a href="#">LandUse Data Dictionary</a> .

### **Class LandUseClassValue**

Definition: LandUseClassValue is a code list used to further classify a LandUse.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.12.4. Class LandUseFunctionValue**

<b>Requirement 199</b>	<a href="#">/req/LandUse/LandUseFunctionValue</a>
A	Any use of the LandUseFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the LandUseFunctionValue UML class as documented in the <a href="#">LandUse Data Dictionary</a> .

### **Class LandUseFunctionValue**

Definition: LandUseFunctionValue is a code list that enumerates the different purposes of a LandUse.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.12.5. Class LandUseUsageValue**

<b>Requirement 200</b>	<a href="#">/req/LandUse/LandUseUsageValue</a>
A	Any use of the LandUseUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the LandUseUsageValue UML class as documented in the <a href="#">LandUse Data Dictionary</a> .

### **Class LandUseUsageValue**

Definition: LandUseUsageValue is a code list that enumerates the different uses of a LandUse.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.12.6. Additional Information**

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

## 11.13. Point Cloud

### Requirements Class

<http://www.opengis.net/spec/CityGML/3.0/req/req-class-pointcloud>

Target type	Conceptual Model
Dependency	<a href="#">/req/req-class-core</a>

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](#).

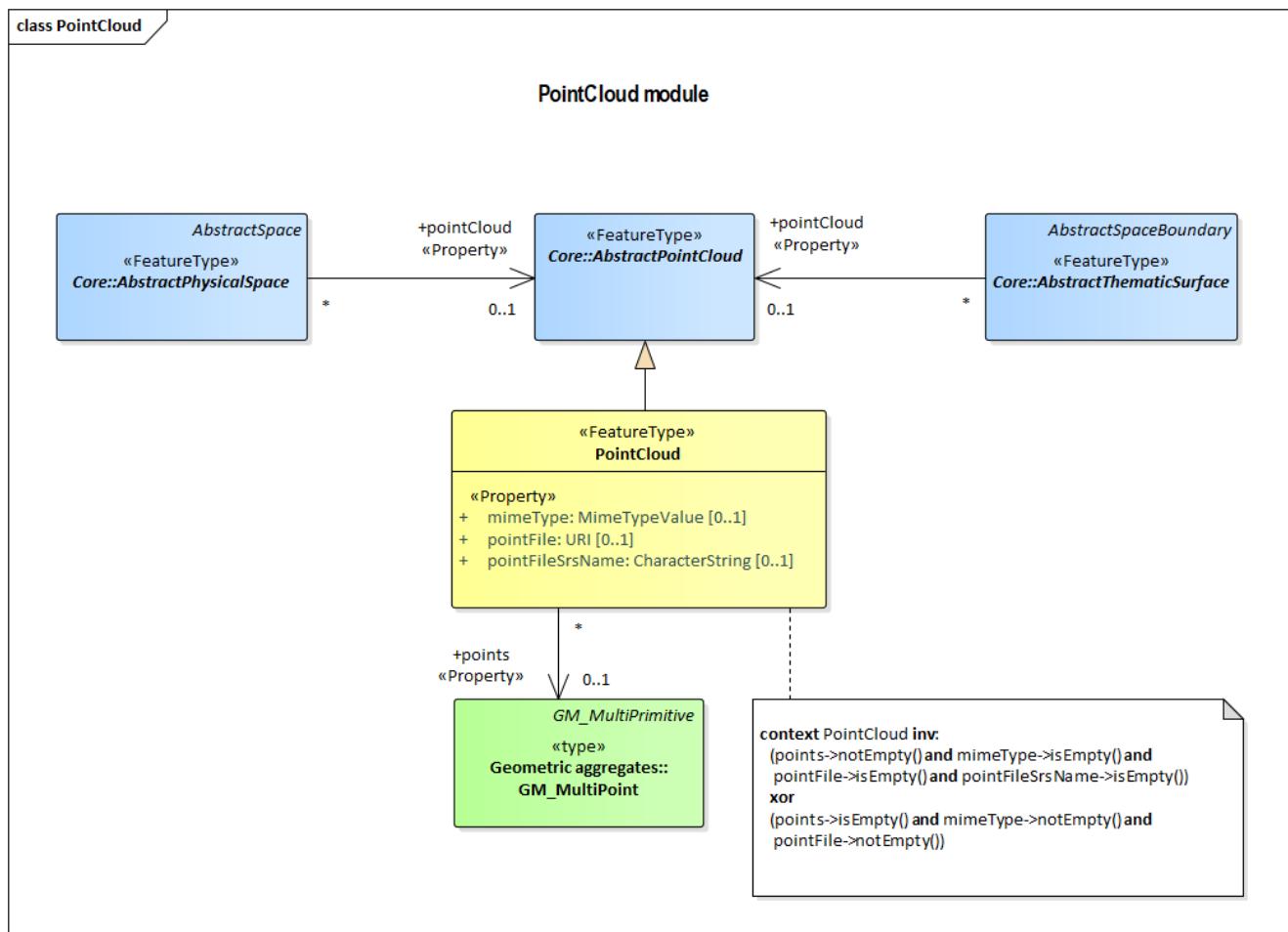


Figure 26. UML diagram of the Point Cloud Model.

### 11.13.1. PointCloud Package

#### Package PointCloud

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

### 11.13.2. Class PointCloud

Requirement 201	/req/PointCloud/PointCloud
A	The Implementation Specification SHALL contain an element with the same definition as the PointCloud UML class as documented in the PointCloud section of the <a href="#">PointCloud Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the PointCloud UML class as documented in the PointCloud section of the <a href="#">PointCloud Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the PointCloud UML class; including the name, definition, type, and cardinality of those documented in the PointCloud section of the <a href="#">PointCloud Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the PointCloud UML class; including the name, definition, type, and cardinality of those documented in the PointCloud section of the <a href="#">PointCloud Data Dictionary</a> .

#### Class 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»

#### Relations

Relation Name: points

Cardinality: 0..1

Target Class: [GM\\_MultiPoint](#)

Definition: Relates to the 3D MultiPoint geometry of the PointCloud stored inline with the city model.

#### Attributes

Attribute Name: mimeType

Value Type: [MimeTypeValue](#)

Definition: Specifies the MIME type of the external point cloud file.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	pointFile
Value Type:	URI
Definition:	Specifies the URI that points to the external point cloud file.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	pointFileSrsName
Value Type:	CharacterString
Definition:	Indicates the coordinate reference system used by the external point cloud file.
Multiplicity:	[0..1]
Stereotype:	«Property»

### 11.13.3. Additional Information

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

## 11.14. Digital Terrain Model

### Requirements Class

<http://www.opengis.net/spec/CityGML/3.1/req/req-class-relief>

Target type	Conceptual Model
Dependency	<a href="/req/req-class-core">/req/req-class-core</a>

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](#).

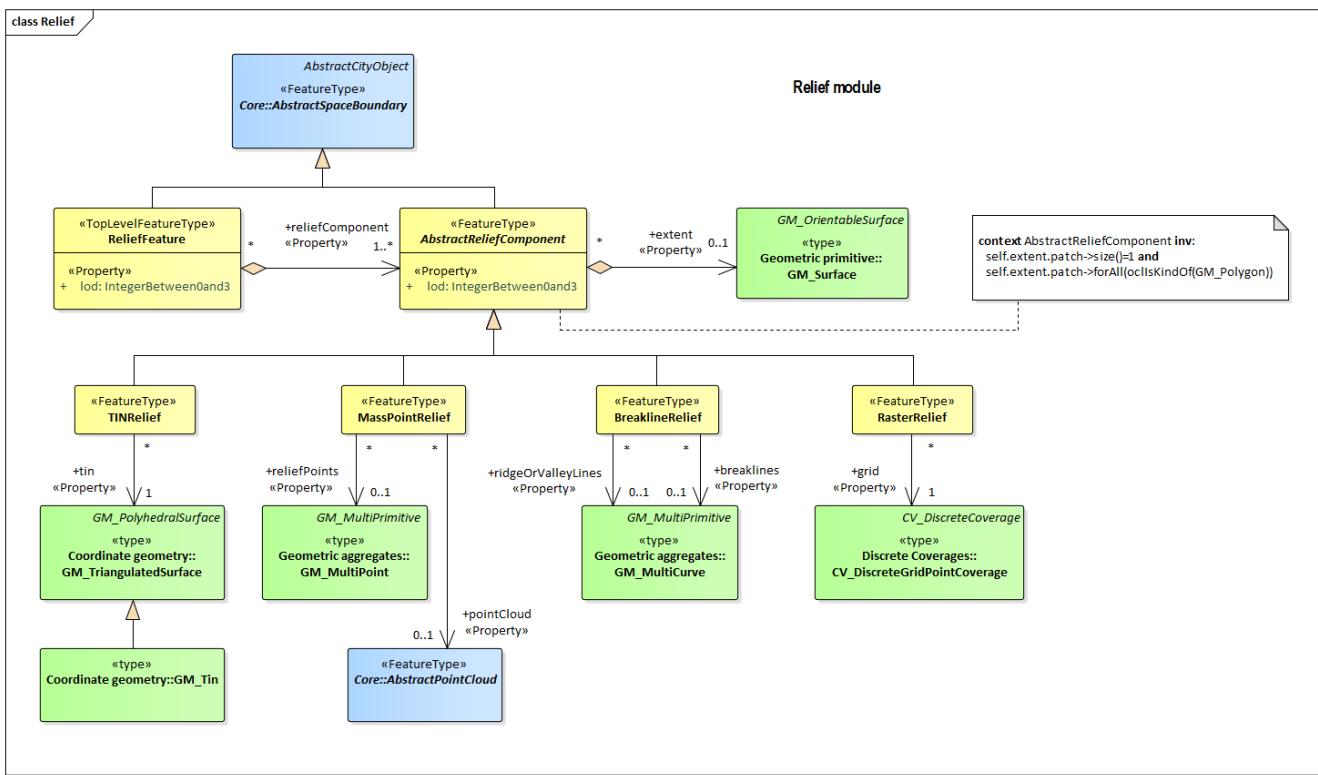


Figure 27. UML diagram of Relief module.

### 11.14.1. Relief 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.

**Stereotype:** «ApplicationSchema»

**Parent Package:** CityGML

### 11.14.2. Class AbstractReliefComponent

Requirement 202	/req/Relief/AbstractReliefComponent
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractReliefComponent UML class as documented in the AbstractReliefComponent section of the <a href="#">Relief Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractReliefComponent UML class as documented in the AbstractReliefComponent section of the <a href="#">Relief Data Dictionary</a> .

C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractReliefComponent UML class; including the name, definition, type, and cardinality of those documented in the AbstractReliefComponent section of the <a href="#">Relief Data Dictionary</a> .
---	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class 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»

### Relations

Relation Name: extent

Cardinality: 0..1

Target Class: [GM\\_Surface](#)

Definition: Indicates the geometrical extent of the terrain component. The geometrical extent is provided as a 2D Surface geometry.

### Attributes

Attribute Name: lod

Value Type: IntegerBetween0and3

Definition: Indicates the Level of Detail of the terrain component.

Multiplicity:

Stereotype: «Property»

### 11.14.3. Class BreaklineRelief

Requirement 203	/req/Relief/BreaklineRelief
A	The Implementation Specification SHALL contain an element with the same definition as the BreaklineRelief UML class as documented in the BreaklineRelief section of the <a href="#">Relief Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BreaklineRelief UML class as documented in the BreaklineRelief section of the <a href="#">Relief Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the BreaklineRelief UML class; including the name, definition, type, and cardinality of those documented in the BreaklineRelief section of the <a href="#">Relief Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the BreaklineRelief UML class; including the name, definition, type, and cardinality of those documented in the BreaklineRelief section of the <a href="#">Relief Data Dictionary</a> .
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class 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»

### Relations

Relation Name: ridgeOrValleyLines

Cardinality: 0..1

Target Class: [GM\\_MultiCurve](#)

Definition: Relates to the 3D MultiCurve geometry of the MassPointRelief. This association role is used to represent ridge or valley lines.

Relation Name: breaklines

Cardinality: 0..1

Target Class: [GM\\_MultiCurve](#)

Definition: Relates to the 3D MultiCurve geometry of the MassPointRelief. This association role is used to represent break lines.

### Attributes

## 11.14.4. Class MassPointRelief

Requirement 204	/req/Relief/MassPointRelief
A	The Implementation Specification SHALL contain an element with the same definition as the MassPointRelief UML class as documented in the MassPointRelief section of the <a href="#">Relief Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the MassPointRelief UML class as documented in the MassPointRelief section of the <a href="#">Relief Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the MassPointRelief UML class; including the name, definition, type, and cardinality of those documented in the MassPointRelief section of the <a href="#">Relief Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the MassPointRelief UML class; including the name, definition, type, and cardinality of those documented in the MassPointRelief section of the <a href="#">Relief Data Dictionary</a> .
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class MassPointRelief

Definition: A MassPointRelief represents a terrain component as a collection of 3D points.

Subclass Of: [AbstractReliefComponent](#)

Stereotype: «FeatureType»

#### Relations

Relation Name: reliefPoints

Cardinality: 0..1

Target Class: [GM\\_MultiPoint](#)

Definition: Relates to the 3D MultiPoint geometry of the MassPointRelief.

Relation Name: pointCloud

Cardinality: 0..1

Target Class: [AbstractPointCloud](#)

Definition: Relates to the 3D PointCloud of the MassPointRelief.

#### Attributes

### 11.14.5. Class RasterRelief

Requirement 205	/req/Relief/RasterRelief
A	The Implementation Specification SHALL contain an element with the same definition as the RasterRelief UML class as documented in the RasterRelief section of the <a href="#">Relief Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the RasterRelief UML class as documented in the RasterRelief section of the <a href="#">Relief Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the RasterRelief UML class; including the name, definition, type, and cardinality of those documented in the RasterRelief section of the <a href="#">Relief Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the RasterRelief UML class; including the name, definition, type, and cardinality of those documented in the RasterRelief section of the <a href="#">Relief Data Dictionary</a> .

## Class RasterRelief

Definition: A RasterRelief represents a terrain component as a regular raster or grid.

Subclass Of: [AbstractReliefComponent](#)

Stereotype: «FeatureType»

### Relations

Relation Name: grid

Cardinality: 1

Target Class: [CV\\_DiscreteGridPointCoverage](#)

Definition: Relates to the DiscreteGridPointCoverage of the RasterRelief.

### Attributes

## 11.14.6. Class ReliefFeature

Requirement 206	/req/Relief/ReliefFeature
A	The Implementation Specification SHALL contain an element with the same definition as the ReliefFeature UML class as documented in the ReliefFeature section of the <a href="#">Relief Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ReliefFeature UML class as documented in the ReliefFeature section of the <a href="#">Relief Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the ReliefFeature UML class; including the name, definition, type, and cardinality of those documented in the ReliefFeature section of the <a href="#">Relief Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ReliefFeature UML class; including the name, definition, type, and cardinality of those documented in the ReliefFeature section of the <a href="#">Relief Data Dictionary</a> .

## Class 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»

### Relations

Relation Name: reliefComponent

Cardinality: 1..\*

Target Class: [AbstractReliefComponent](#)

Definition: Relates to the terrain components that are part of the ReliefFeature.

### Attributes

Attribute Name:	lod
Value Type:	IntegerBetween0and3
Definition:	Indicates the Level of Detail of the ReliefFeature.
Multiplicity:	
Stereotype:	«Property»

## 11.14.7. Class TINRelief

Requirement 207	/req/Relief/TINRelief
A	The Implementation Specification SHALL contain an element with the same definition as the TINRelief UML class as documented in the TINRelief section of the <a href="#">Relief Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TINRelief UML class as documented in the TINRelief section of the <a href="#">Relief Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TINRelief UML class; including the name, definition, type, and cardinality of those documented in the TINRelief section of the <a href="#">Relief Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TINRelief UML class; including the name, definition, type, and cardinality of those documented in the TINRelief section of the <a href="#">Relief Data Dictionary</a> .

### Class TINRelief

Definition: A TINRelief represents a terrain component as a triangulated irregular network.  
 Subclass Of: [AbstractReliefComponent](#)  
 Stereotype: «FeatureType»

### Relations

Relation Name: tin  
 Cardinality: 1  
 Target Class: [GM\\_TriangulatedSurface](#)  
 Definition: Relates to the triangulated surface of the TINRelief.

### Attributes

## 11.14.8. Additional Information

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

## 11.15. Transportation

### Requirements Class

<http://www.opengis.net/spec/CityGML/3.0/req/req-class-transportation>

Target type	Conceptual Model
Dependency	<a href="#">/req/req-class-core</a>

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](#).

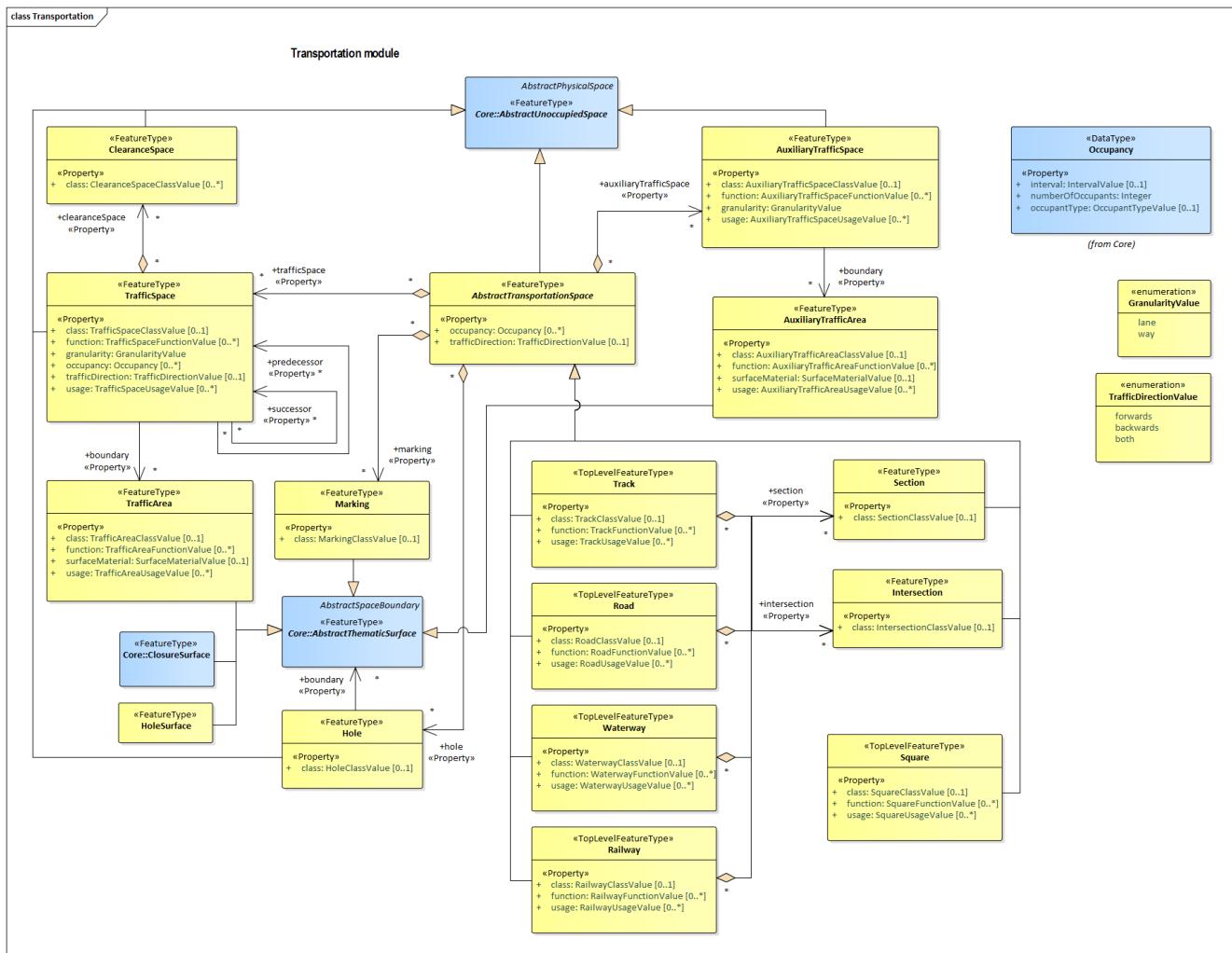


Figure 28. UML diagram of the Transportation Model.

### 11.15.1. Transportation 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.

**Stereotype:** «ApplicationSchema»

**Parent Package:** CityGML

### 11.15.2. Class AbstractTransportationSpace

Requirement 208	/req/Transportation/AbstractTransportationSpace
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the <b>AbstractTransportationSpace</b> UML class as documented in the <b>AbstractTransportationSpace</b> section of the <a href="#">Transportation Data Dictionary</a> .

B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractTransportationSpace UML class as documented in the AbstractTransportationSpace section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractTransportationSpace UML class; including the name, definition, type, and cardinality of those documented in the AbstractTransportationSpace section of the <a href="#">Transportation Data Dictionary</a> .

### Class AbstractTransportationSpace

Definition: AbstractTransportationSpace is the abstract superclass of transportation objects such as Roads, Tracks, Railways, Waterways or Squares.

Subclass Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

### Relations

Relation Name: trafficSpace

Cardinality: \*

Target Class: [TrafficSpace](#)

Definition: Relates to the traffic spaces that are part of the transportation space.

Relation Name: hole

Cardinality: \*

Target Class: [Hole](#)

Definition: Relates to the holes that are part of the transportation space.

Relation Name: auxiliaryTrafficSpace

Cardinality: \*

Target Class: [AuxiliaryTrafficSpace](#)

Definition: Relates to the auxiliary traffic spaces that are part of the transportation space.

Relation Name: marking

Cardinality: \*

Target Class: [Marking](#)

Definition: Relates to the markings that are part of the transportation space.

### Attributes

Attribute Name: occupancy

Value Type: Occupancy

Definition: Provides information on the residency of persons, vehicles, or other moving features in the transportation space.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name:	trafficDirection
Value Type:	TrafficDirectionValue
Definition:	Indicates the direction of traffic flow relative to the corresponding linear geometry representation.
Multiplicity:	[0..1]
Stereotype:	«Property»

### 11.15.3. Class AuxiliaryTrafficArea

Requirement 209	/req/Transportation/AuxiliaryTrafficArea
A	The Implementation Specification SHALL contain an element with the same definition as the AuxiliaryTrafficArea UML class as documented in the AuxiliaryTrafficArea section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the AuxiliaryTrafficArea UML class as documented in the AuxiliaryTrafficArea section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the AuxiliaryTrafficArea UML class; including the name, definition, type, and cardinality of those documented in the AuxiliaryTrafficArea section of the <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the AuxiliaryTrafficArea UML class; including the name, definition, type, and cardinality of those documented in the AuxiliaryTrafficArea section of the <a href="#">Transportation Data Dictionary</a> .

#### Class AuxiliaryTrafficArea

Definition:	An AuxiliaryTrafficArea is the ground surface of an AuxiliaryTrafficSpace.
Subclass Of:	<a href="#">AbstractThematicSurface</a>
Stereotype:	«FeatureType»

#### Relations

#### Attributes

Attribute Name:	class
Value Type:	AuxiliaryTrafficAreaClassValue
Definition:	Indicates the specific type of the AuxiliaryTrafficArea.
Multiplicity:	[0..1]
Stereotype:	«Property»

Attribute Name:	function
Value Type:	AuxiliaryTrafficAreaFunctionValue
Definition:	Specifies the intended purposes of the AuxiliaryTrafficArea.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	surfaceMaterial
Value Type:	SurfaceMaterialValue
Definition:	Specifies the type of pavement of the AuxiliaryTrafficArea.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	AuxiliaryTrafficAreaUsageValue
Definition:	Specifies the actual uses of the AuxiliaryTrafficArea.
Multiplicity:	[0..*]
Stereotype:	«Property»

#### 11.15.4. Class AuxiliaryTrafficAreaClassValue

Requirement 210	/req/Transportation/AuxiliaryTrafficAreaClassValue
A	Any use of the AuxiliaryTrafficAreaClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuxiliaryTrafficAreaClassValue UML class as documented in the AuxiliaryTrafficAreaClassValue section of the <a href="#">Transportation Data Dictionary</a> .

##### Class AuxiliaryTrafficAreaClassValue

Definition: AuxiliaryTrafficAreaClassValue is a code list used to further classify an AuxiliaryTrafficArea.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

##### Relations

##### Attributes

#### 11.15.5. Class AuxiliaryTrafficAreaFunctionValue

Requirement 211	/req/Transportation/AuxiliaryTrafficAreaFunctionValue
A	Any use of the AuxiliaryTrafficAreaFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuxiliaryTrafficAreaFunctionValue UML class as documented in the AuxiliaryTrafficAreaFunctionValue section of the <a href="#">Transportation Data Dictionary</a> .

### **Class AuxiliaryTrafficAreaFunctionValue**

Definition: AuxiliaryTrafficAreaFunctionValue is a code list that enumerates the different purposes of an AuxiliaryTrafficArea.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.15.6. Class AuxiliaryTrafficAreaUsageValue**

<b>Requirement 212</b>	<b>/req/Transportation/AuxiliaryTrafficAreaUsageValue</b>
A	Any use of the AuxiliaryTrafficAreaUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuxiliaryTrafficAreaUsageValue UML class as documented in the AuxiliaryTrafficAreaUsageValue section of the <a href="#">Transportation Data Dictionary</a> .

### **Class AuxiliaryTrafficAreaUsageValue**

Definition: AuxiliaryTrafficAreaUsageValue is a code list that enumerates the different uses of an AuxiliaryTrafficArea.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.15.7. Class AuxiliaryTrafficSpace**

<b>Requirement 213</b>	<b>/req/Transportation/AuxiliaryTrafficSpace</b>
A	The Implementation Specification SHALL contain an element with the same definition as the AuxiliaryTrafficSpace UML class as documented in the AuxiliaryTrafficSpace section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the AuxiliaryTrafficSpace UML class as documented in the AuxiliaryTrafficSpace section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the AuxiliaryTrafficSpace UML class; including the name, definition, type, and cardinality of those documented in the AuxiliaryTrafficSpace section of the <a href="#">Transportation Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the AuxiliaryTrafficSpace UML class; including the name, definition, type, and cardinality of those documented in the AuxiliaryTrafficSpace section of the <a href="#">Transportation Data Dictionary</a> .
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class AuxiliaryTrafficSpace

Definition: An AuxiliaryTrafficSpace is a space within the transportation space not intended for traffic purposes.

Subclass Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

### Relations

Relation Name: boundary

Cardinality: \*

Target Class: [AuxiliaryTrafficArea](#)

Definition:

### Attributes

Attribute Name: class

Value Type: AuxiliaryTrafficSpaceClassName

Definition: Indicates the specific type of the AuxiliaryTrafficSpace.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: AuxiliaryTrafficSpaceFunctionName

Definition: Specifies the intended purposes of the AuxiliaryTrafficSpace.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: granularity

Value Type: GranularityValue

Definition: Defines whether auxiliary traffic spaces are represented by individual ways or by individual lanes, depending on the desired level of detail.

Multiplicity:

Stereotype: «Property»

Attribute Name: usage

Value Type: AuxiliaryTrafficSpaceUsageName

Definition: Specifies the actual uses of the AuxiliaryTrafficSpace.

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.15.8. Class AuxiliaryTrafficSpaceClassName

Requirement 214	/req/Transportation/AuxiliaryTrafficSpaceClassName
-----------------	----------------------------------------------------

A	Any use of the AuxiliaryTrafficSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuxiliaryTrafficSpaceClassValue UML class as documented in the AuxiliaryTrafficSpaceClassValue section of the <a href="#">Transportation Data Dictionary</a> .
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class AuxiliaryTrafficSpaceClassValue

Definition: AuxiliaryTrafficSpaceClassValue is a code list used to further classify an AuxiliaryTrafficSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.15.9. Class AuxiliaryTrafficSpaceFunctionValue

Requirement 215	/req/Transportation/AuxiliaryTrafficSpaceFunctionValue
A	Any use of the AuxiliaryTrafficSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuxiliaryTrafficSpaceFunctionValue UML class as documented in the AuxiliaryTrafficSpaceFunctionValue section of the <a href="#">Transportation Data Dictionary</a> .

### Class AuxiliaryTrafficSpaceFunctionValue

Definition: AuxiliaryTrafficSpaceFunctionValue is a code list that enumerates the different purposes of an AuxiliaryTrafficSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.15.10. Class AuxiliaryTrafficSpaceUsageValue

Requirement 216	/req/Transportation/AuxiliaryTrafficSpaceUsageValue
A	Any use of the AuxiliaryTrafficSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuxiliaryTrafficSpaceUsageValue UML class as documented in the AuxiliaryTrafficSpaceUsageValue section of the <a href="#">Transportation Data Dictionary</a> .

### **Class AuxiliaryTrafficSpaceUsageValue**

Definition: AuxiliaryTrafficSpaceUsageValue is a code list that enumerates the different uses of an AuxiliaryTrafficSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.15.11. Class ClearanceSpace**

Requirement 217	/req/Transportation/ClearanceSpace
A	The Implementation Specification SHALL contain an element with the same definition as the ClearanceSpace UML class as documented in the ClearanceSpace section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ClearanceSpace UML class as documented in the ClearanceSpace section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the ClearanceSpace UML class; including the name, definition, type, and cardinality of those documented in the ClearanceSpace section of the <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ClearanceSpace UML class; including the name, definition, type, and cardinality of those documented in the ClearanceSpace section of the <a href="#">Transportation Data Dictionary</a> .

### **Class 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»

#### **Relations**

#### **Attributes**

Attribute Name: class

Value Type: ClearanceSpaceClassValue

Definition: Indicates the specific type of the ClearanceSpace.

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.15.12. Class ClearanceSpaceClassValue

Requirement 218	/req/Transportation/ClearanceSpaceClassValue
A	Any use of the ClearanceSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the ClearanceSpaceClassValue UML class as documented in the ClearanceSpaceClassValue section of the <a href="#">Transportation Data Dictionary</a> .

### Class ClearanceSpaceClassValue

Definition: ClearanceSpaceClassValue is a code list used to further classify a ClearanceSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Relations

### Attributes

## 11.15.13. Class Hole

Requirement 219	/req/Transportation/Hole
A	The Implementation Specification SHALL contain an element with the same definition as the Hole UML class as documented in the Hole section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Hole UML class as documented in the Hole section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Hole UML class; including the name, definition, type, and cardinality of those documented in the Hole section of the <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Hole UML class; including the name, definition, type, and cardinality of those documented in the Hole section of the <a href="#">Transportation Data Dictionary</a> .

### Class 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»

### Relations

Relation Name: boundary  
 Cardinality: \*  
 Target Class: [AbstractThematicSurface](#)  
 Definition:

### Attributes

Attribute Name: class  
 Value Type: HoleClassValue  
 Definition: Indicates the specific type of the Hole.  
 Multiplicity: [0..1]  
 Stereotype: «Property»

## 11.15.14. Class HoleClassValue

Requirement 220	/req/Transportation/HoleClassValue
A	Any use of the HoleClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the HoleClassValue UML class as documented in the HoleClassValue section of the <a href="#">Transportation Data Dictionary</a> .

### Class HoleClassValue

Definition: HoleClassValue is a code list used to further classify a Hole.  
 Subclass Of: <-- section,>>  
 Stereotype: «CodeList»

### Relations

### Attributes

## 11.15.15. Class HoleSurface

Requirement 221	/req/Transportation/HoleSurface
A	The Implementation Specification SHALL contain an element with the same definition as the HoleSurface UML class as documented in the HoleSurface section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the HoleSurface UML class as documented in the HoleSurface section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the HoleSurface UML class; including the name, definition, type, and cardinality of those documented in the HoleSurface section of the <a href="#">Transportation Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the HoleSurface UML class; including the name, definition, type, and cardinality of those documented in the HoleSurface section of the <a href="#">Transportation Data Dictionary</a> .
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class HoleSurface

Definition: A HoleSurface is a representation of the ground surface of a hole.

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

### Relations

### Attributes

## 11.15.16. Class Intersection

Requirement 222	/req/Transportation/Intersection
A	The Implementation Specification SHALL contain an element with the same definition as the Intersection UML class as documented in the Intersection section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Intersection UML class as documented in the Intersection section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Intersection UML class; including the name, definition, type, and cardinality of those documented in the Intersection section of the <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Intersection UML class; including the name, definition, type, and cardinality of those documented in the Intersection section of the <a href="#">Transportation Data Dictionary</a> .

### Class 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»

### Relations

### Attributes

Attribute Name:	class
Value Type:	IntersectionClassValue
Definition:	Indicates the specific type of the Intersection.
Multiplicity:	[0..1]
Stereotype:	«Property»

### 11.15.17. Class IntersectionClassValue

Requirement 223	/req/Transportation/IntersectionClassValue
A	Any use of the IntersectionClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the IntersectionClassValue UML class as documented in the IntersectionClassValue section of the <a href="#">Transportation Data Dictionary</a> .

#### Class IntersectionClassValue

Definition:	IntersectionClassValue is a code list used to further classify an Intersection.
Subclass Of:	<-- section,>>
Stereotype:	«CodeList»

#### Relations

#### Attributes

### 11.15.18. Class Marking

Requirement 224	/req/Transportation/Marking
A	The Implementation Specification SHALL contain an element with the same definition as the Marking UML class as documented in the Marking section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Marking UML class as documented in the Marking section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Marking UML class; including the name, definition, type, and cardinality of those documented in the Marking section of the <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Marking UML class; including the name, definition, type, and cardinality of those documented in the Marking section of the <a href="#">Transportation Data Dictionary</a> .

## **Class 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»

## **Relations**

## **Attributes**

Attribute Name: class

Value Type: [MarkingClassValue](#)

Definition: Indicates the specific type of the Marking.

Multiplicity: [0..1]

Stereotype: «Property»

## **11.15.19. Class MarkingClassValue**

Requirement 225 /req/Transportation/MarkingClassValue	
A	Any use of the MarkingClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the MarkingClassValue UML class as documented in the MarkingClassValue section of the <a href="#">Transportation Data Dictionary</a> .

## **Class MarkingClassValue**

Definition: MarkingClassValue is a code list used to further classify a Marking.

Subclass Of: <-- section,-->

Stereotype: «CodeList»

## **Relations**

## **Attributes**

## **11.15.20. Class Railway**

Requirement 226 /req/Transportation/Railway	
A	The Implementation Specification SHALL contain an element with the same definition as the Railway UML class as documented in the Railway section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Railway UML class as documented in the Railway section of the <a href="#">Transportation Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the Railway UML class; including the name, definition, type, and cardinality of those documented in the Railway section of the <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Railway UML class; including the name, definition, type, and cardinality of those documented in the Railway section of the <a href="#">Transportation Data Dictionary</a> .

### Class Railway

Definition: A Railway is a transportation space used by wheeled vehicles on rails.

Subclass Of: [AbstractTransportationSpace](#)

Stereotype: «TopLevelFeatureType»

### Relations

Relation Name: section

Cardinality: \*

Target Class: [Section](#)

Definition: Relates to the sections that are part of the Railway.

Relation Name: intersection

Cardinality: \*

Target Class: [Intersection](#)

Definition: Relates to the intersections that are part of the Railway.

### Attributes

Attribute Name: class

Value Type: [RailwayClassValue](#)

Definition: Indicates the specific type of the Railway.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: [RailwayFunctionValue](#)

Definition: Specifies the intended purposes of the Railway.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: [RailwayUsageValue](#)

Definition: Specifies the actual uses of the Railway.

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.15.21. Class RailwayClassValue

Requirement 227	/req/Transportation/RailwayClassValue
-----------------	---------------------------------------

A	Any use of the RailwayClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RailwayClassValue UML class as documented in the RailwayClassValue section of the <a href="#">Transportation Data Dictionary</a> .
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class RailwayClassValue

Definition: RailwayClassValue is a code list used to further classify a Railway.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Relations

### Attributes

## 11.15.22. Class RailwayFunctionValue

Requirement 228	/req/Transportation/RailwayFunctionValue
A	Any use of the RailwayFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RailwayFunctionValue UML class as documented in the RailwayFunctionValue section of the <a href="#">Transportation Data Dictionary</a> .

### Class RailwayFunctionValue

Definition: RailwayFunctionValue is a code list that enumerates the different purposes of a Railway.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Relations

### Attributes

## 11.15.23. Class RailwayUsageValue

Requirement 229	/req/Transportation/RailwayUsageValue
A	Any use of the RailwayUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RailwayUsageValue UML class as documented in the RailwayUsageValue section of the <a href="#">Transportation Data Dictionary</a> .

### Class RailwayUsageValue

Definition: RailwayUsageValue is a code list that enumerates the different uses of a Railway.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Relations

## Attributes

### 11.15.24. Class Road

Requirement 230	/req/Transportation/Road
A	The Implementation Specification SHALL contain an element with the same definition as the Road UML class as documented in the Road section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Road UML class as documented in the Road section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Road UML class; including the name, definition, type, and cardinality of those documented in the Road section of the <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Road UML class; including the name, definition, type, and cardinality of those documented in the Road section of the <a href="#">Transportation Data Dictionary</a> .

## Class Road

Definition: A Road is a transportation space used by vehicles, bicycles and/or pedestrians.

Subclass Of: [AbstractTransportationSpace](#)

Stereotype: «TopLevelFeatureType»

## Relations

Relation Name: intersection

Cardinality: \*

Target Class: [Intersection](#)

Definition: Relates to the intersections that are part of the Road.

Relation Name: section

Cardinality: \*

Target Class: [Section](#)

Definition: Relates to the sections that are part of the Road.

## Attributes

Attribute Name: class

Value Type: [RoadClassValue](#)

Definition: Indicates the specific type of the Road.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	function
Value Type:	RoadFunctionValue
Definition:	Specifies the intended purposes of the Road.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	RoadUsageValue
Definition:	Specifies the actual uses of the Road.
Multiplicity:	[0..*]
Stereotype:	«Property»

## 11.15.25. Class RoadClassValue

Requirement 231	/req/Transportation/RoadClassValue
A	Any use of the RoadClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RoadClassValue UML class as documented in the <a href="#">Transportation Data Dictionary</a> .

### Class RoadClassValue

Definition: RoadClassValue is a code list used to further classify a Road.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Relations

### Attributes

## 11.15.26. Class RoadFunctionValue

Requirement 232	/req/Transportation/RoadFunctionValue
A	Any use of the RoadFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RoadFunctionValue UML class as documented in the <a href="#">Transportation Data Dictionary</a> .

### Class RoadFunctionValue

Definition: RoadFunctionValue is a code list that enumerates the different purposes of a Road.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Relations

### Attributes

## 11.15.27. Class RoadUsageValue

Requirement 233	/req/Transportation/RoadUsageValue
A	Any use of the RoadUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RoadUsageValue UML class as documented in the RoadUsageValue section of the <a href="#">Transportation Data Dictionary</a> .

### Class RoadUsageValue

Definition: RoadUsageValue is a code list that enumerates the different uses of a Road.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Relations

### Attributes

## 11.15.28. Class Section

Requirement 234	/req/Transportation/Section
A	The Implementation Specification SHALL contain an element with the same definition as the Section UML class as documented in the Section section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Section UML class as documented in the Section section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Section UML class; including the name, definition, type, and cardinality of those documented in the Section section of the <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Section UML class; including the name, definition, type, and cardinality of those documented in the Section section of the <a href="#">Transportation Data Dictionary</a> .

### Class Section

Definition: A Section is a transportation space that is a segment of a Road, Railway, Track, or Waterway.

Subclass Of: [AbstractTransportationSpace](#)

Stereotype: «FeatureType»

### Relations

### Attributes

Attribute Name:	class
Value Type:	SectionClassValue
Definition:	Indicates the specific type of the Section.
Multiplicity:	[0..1]
Stereotype:	«Property»

### 11.15.29. Class SectionClassValue

Requirement 235	/req/Transportation/SectionClassValue
A	Any use of the SectionClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SectionClassValue UML class as documented in the SectionClassValue section of the <a href="#">Transportation Data Dictionary</a> .

#### Class SectionClassValue

Definition: SectionClassValue is a code list used to further classify a Section.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.15.30. Class Square

Requirement 236	/req/Transportation/Square
A	The Implementation Specification SHALL contain an element with the same definition as the Square UML class as documented in the Square section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Square UML class as documented in the Square section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Square UML class; including the name, definition, type, and cardinality of those documented in the Square section of the <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Square UML class; including the name, definition, type, and cardinality of those documented in the Square section of the <a href="#">Transportation Data Dictionary</a> .

## Class 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»

## Relations

## Attributes

Attribute Name: class

Value Type: SquareClassValue

Definition: Indicates the specific type of the Square.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: SquareFunctionValue

Definition: Specifies the intended purposes of the Square.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: SquareUsageValue

Definition: Specifies the actual uses of the Square.

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.15.31. Class SquareClassValue

Requirement 237	/req/Transportation/SquareClassValue
A	Any use of the SquareClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SquareClassValue UML class as documented in the <a href="#">Transportation Data Dictionary</a> .

## Class SquareClassValue

Definition: SquareClassValue is a code list used to further classify a Square.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

## Relations

## Attributes

## 11.15.32. Class SquareFunctionValue

Requirement 238	/req/Transportation/SquareFunctionValue
-----------------	-----------------------------------------

A	Any use of the SquareFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SquareFunctionValue UML class as documented in the SquareFunctionValue section of the <a href="#">Transportation Data Dictionary</a> .
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class SquareFunctionValue

Definition: SquareFunctionValue is a code list that enumerates the different purposes of a Square.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.15.33. Class SquareUsageValue

Requirement 239	/req/Transportation/SquareUsageValue
A	Any use of the SquareUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SquareUsageValue UML class as documented in the SquareUsageValue section of the <a href="#">Transportation Data Dictionary</a> .

### Class SquareUsageValue

Definition: SquareUsageValue is a code list that enumerates the different uses of a Square.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.15.34. Class SurfaceMaterialValue

Requirement 240	/req/Transportation/SurfaceMaterialValue
A	Any use of the SurfaceMaterialValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SurfaceMaterialValue UML class as documented in the SurfaceMaterialValue section of the <a href="#">Transportation Data Dictionary</a> .

### Class SurfaceMaterialValue

Definition: SurfaceMaterialValue is a code list that enumerates the different surface materials.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

## Attributes

### 11.15.35. Class Track

Requirement 241	/req/Transportation/Track
A	The Implementation Specification SHALL contain an element with the same definition as the Track UML class as documented in the Track section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Track UML class as documented in the Track section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Track UML class; including the name, definition, type, and cardinality of those documented in the Track section of the <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Track UML class; including the name, definition, type, and cardinality of those documented in the Track section of the <a href="#">Transportation Data Dictionary</a> .

## Class 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»

## Relations

Relation Name: section  
Cardinality: \*  
Target Class: [Section](#)  
Definition: Relates to the sections that are part of the Track.

Relation Name: intersection  
Cardinality: \*  
Target Class: [Intersection](#)  
Definition: Relates to the intersections that are part of the Track.

## Attributes

Attribute Name: class  
Value Type: [TrackClassValue](#)  
Definition: Indicates the specific type of the Track.  
Multiplicity: [0..1]  
Stereotype: «Property»

Attribute Name:	function
Value Type:	TrackFunctionValue
Definition:	Specifies the intended purposes of the Track.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	TrackUsageValue
Definition:	Specifies the actual uses of the Track.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.15.36. Class TrackClassValue

Requirement 242	/req/Transportation/TrackClassValue
A	Any use of the TrackClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrackClassValue UML class as documented in the TrackClassValue section of the <a href="#">Transportation Data Dictionary</a> .

#### Class TrackClassValue

Definition: TrackClassValue is a code list used to further classify a Track.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.15.37. Class TrackFunctionValue

Requirement 243	/req/Transportation/TrackFunctionValue
A	Any use of the TrackFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrackFunctionValue UML class as documented in the TrackFunctionValue section of the <a href="#">Transportation Data Dictionary</a> .

#### Class TrackFunctionValue

Definition: TrackFunctionValue is a code list that enumerates the different purposes of a Track.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.15.38. Class TrackUsageValue

Requirement 244	/req/Transportation/TrackUsageValue
A	Any use of the TrackUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrackUsageValue UML class as documented in the TrackUsageValue section of the <a href="#">Transportation Data Dictionary</a> .

#### Class TrackUsageValue

Definition: TrackUsageValue is a code list that enumerates the different uses of a Track.

Subclass Of: <-- section,-->

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.15.39. Class TrafficArea

Requirement 245	/req/Transportation/TrafficArea
A	The Implementation Specification SHALL contain an element with the same definition as the TrafficArea UML class as documented in the TrafficArea section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TrafficArea UML class as documented in the TrafficArea section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TrafficArea UML class; including the name, definition, type, and cardinality of those documented in the TrafficArea section of the <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TrafficArea UML class; including the name, definition, type, and cardinality of those documented in the TrafficArea section of the <a href="#">Transportation Data Dictionary</a> .

#### Class 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»

#### Relations

#### Attributes

Attribute Name:	class
Value Type:	TrafficAreaClassValue
Definition:	Indicates the specific type of the TrafficArea.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	TrafficAreaFunctionValue
Definition:	Specifies the intended purposes of the TrafficArea.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	surfaceMaterial
Value Type:	SurfaceMaterialValue
Definition:	Specifies the type of pavement of the TrafficArea.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	TrafficAreaUsageValue
Definition:	Specifies the actual uses of the TrafficArea.
Multiplicity:	[0..*]
Stereotype:	«Property»

#### 11.15.40. Class TrafficAreaClassValue

Requirement 246	/req/Transportation/TrafficAreaClassValue
A	Any use of the TrafficAreaClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficAreaClassValue UML class as documented in the <a href="#">Transportation Data Dictionary</a> .

#### Class TrafficAreaClassValue

Definition: TrafficAreaClassValue is a code list used to further classify a TrafficArea.  
 Subclass Of: <-- section,>>  
 Stereotype: «CodeList»

#### Relations

#### Attributes

#### 11.15.41. Class TrafficAreaFunctionValue

Requirement 247	/req/Transportation/TrafficAreaFunctionValue
-----------------	----------------------------------------------

A	Any use of the TrafficAreaFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficAreaFunctionValue UML class as documented in the TrafficAreaFunctionValue section of the <a href="#">Transportation Data Dictionary</a> .
---	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class TrafficAreaFunctionValue

Definition: TrafficAreaFunctionValue is a code list that enumerates the different purposes of a TrafficArea.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.15.42. Class TrafficAreaUsageValue

Requirement 248	/req/Transportation/TrafficAreaUsageValue
A	Any use of the TrafficAreaUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficAreaUsageValue UML class as documented in the TrafficAreaUsageValue section of the <a href="#">Transportation Data Dictionary</a> .

### Class TrafficAreaUsageValue

Definition: TrafficAreaUsageValue is a code list that enumerates the different uses of a TrafficArea.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.15.43. Class TrafficSpace

Requirement 249	/req/Transportation/TrafficSpace
A	The Implementation Specification SHALL contain an element with the same definition as the TrafficSpace UML class as documented in the TrafficSpace section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TrafficSpace UML class as documented in the TrafficSpace section of the <a href="#">Transportation Data Dictionary</a> .

C	The implementation Specification SHALL represent the attributes of the TrafficSpace UML class; including the name, definition, type, and cardinality of those documented in the TrafficSpace section of the <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TrafficSpace UML class; including the name, definition, type, and cardinality of those documented in the TrafficSpace section of the <a href="#">Transportation Data Dictionary</a> .

## Class 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»

## Relations

Relation Name: clearanceSpace

Cardinality: \*

Target Class: [ClearanceSpace](#)

Definition: Relates to the clearance spaces that are part of the TrafficSpace.

Relation Name: successor

Cardinality: \*

Target Class: [TrafficSpace](#)

Definition: Indicates the successor(s) of the TrafficSpace.

Relation Name: boundary

Cardinality: \*

Target Class: [TrafficArea](#)

Definition:

Relation Name: predecessor

Cardinality: \*

Target Class: [TrafficSpace](#)

Definition: Indicates the predecessor(s) of the TrafficSpace.

## Attributes

Attribute Name: class

Value Type: TrafficSpaceClassValue

Definition: Indicates the specific type of the TrafficSpace.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: TrafficSpaceFunctionValue

Definition: Specifies the intended purposes of the TrafficSpace.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name:	granularity
Value Type:	GranularityValue
Definition:	Defines whether traffic spaces are represented by individual ways or by individual lanes, depending on the desired level of detail.
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	occupancy
Value Type:	Occupancy
Definition:	Provides information on the residency of persons, vehicles, or other moving features in the TrafficSpace.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	trafficDirection
Value Type:	TrafficDirectionValue
Definition:	Indicates the direction of traffic flow relative to the corresponding linear geometry representation.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	TrafficSpaceUsageValue
Definition:	Specifies the actual uses of the TrafficSpace.
Multiplicity:	[0..*]
Stereotype:	«Property»

#### 11.15.44. Class TrafficSpaceClassValue

Requirement 250	/req/Transportation/TrafficSpaceClassValue
A	Any use of the TrafficSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficSpaceClassValue UML class as documented in the TrafficSpaceClassValue section of the <a href="#">Transportation Data Dictionary</a> .

#### Class TrafficSpaceClassValue

Definition: TrafficSpaceClassValue is a code list used to further classify a TrafficSpace.  
 Subclass Of: <-- section,>>  
 StereoType: «CodeList»

#### Relations

#### Attributes

#### 11.15.45. Class TrafficSpaceFunctionValue

Requirement 251	/req/Transportation/TrafficSpaceFunctionValue
-----------------	-----------------------------------------------

A	Any use of the TrafficSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficSpaceFunctionValue UML class as documented in the TrafficSpaceFunctionValue section of the <a href="#">Transportation Data Dictionary</a> .
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class TrafficSpaceFunctionValue

Definition: TrafficSpaceFunctionValue is a code list that enumerates the different purposes of a TrafficSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.15.46. Class TrafficSpaceUsageValue

Requirement 252	/req/Transportation/TrafficSpaceUsageValue
A	Any use of the TrafficSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficSpaceUsageValue UML class as documented in the TrafficSpaceUsageValue section of the <a href="#">Transportation Data Dictionary</a> .

### Class TrafficSpaceUsageValue

Definition: TrafficSpaceUsageValue is a code list that enumerates the different uses of a TrafficSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.15.47. Class TransportationSpaceClassValue

Requirement 253	/req/Transportation/TransportationSpaceClassValue
A	Any use of the TransportationSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TransportationSpaceClassValue UML class as documented in the TransportationSpaceClassValue section of the <a href="#">Transportation Data Dictionary</a> .

### **Class TransportationSpaceClassValue**

Definition: TransportationSpaceClassValue is a code list used to further classify a TransportationSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.15.48. Class TransportationSpaceFunctionValue**

<b>Requirement 254</b>	<b>/req/Transportation/TransportationSpaceFunctionValue</b>
A	Any use of the TransportationSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TransportationSpaceFunctionValue UML class as documented in the TransportationSpaceFunctionValue section of the <a href="#">Transportation Data Dictionary</a> .

### **Class TransportationSpaceFunctionValue**

Definition: TransportationSpaceFunctionValue is a code list that enumerates the different purposes of a TransportationSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.15.49. Class TransportationSpaceUsageValue**

<b>Requirement 255</b>	<b>/req/Transportation/TransportationSpaceUsageValue</b>
A	Any use of the TransportationSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TransportationSpaceUsageValue UML class as documented in the TransportationSpaceUsageValue section of the <a href="#">Transportation Data Dictionary</a> .

### **Class TransportationSpaceUsageValue**

Definition: TransportationSpaceUsageValue is a code list that enumerates the different uses of a TransportationSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

## 11.15.50. Class Waterway

Requirement 256	/req/Transportation/Waterway
A	The Implementation Specification SHALL contain an element with the same definition as the Waterway UML class as documented in the Waterway section of the <a href="#">Transportation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Waterway UML class as documented in the Waterway section of the <a href="#">Transportation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Waterway UML class; including the name, definition, type, and cardinality of those documented in the Waterway section of the <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Waterway UML class; including the name, definition, type, and cardinality of those documented in the Waterway section of the <a href="#">Transportation Data Dictionary</a> .

### Class 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»

### Relations

Relation Name: section

Cardinality: \*

Target Class: [Section](#)

Definition: Relates to the sections that are part of the Waterway.

Relation Name: intersection

Cardinality: \*

Target Class: [Intersection](#)

Definition: Relates to the intersections that are part of the Waterway.

### Attributes

Attribute Name: class

Value Type: [WaterwayClassValue](#)

Definition: Indicates the specific type of the Waterway.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	function
Value Type:	WaterwayFunctionValue
Definition:	Specifies the intended purposes of the Waterway.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	WaterwayUsageValue
Definition:	Specifies the actual uses of the Waterway.
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.15.51. Class WaterwayClassValue

Requirement 257	/req/Transportation/WaterwayClassValue
A	Any use of the WaterwayClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterwayClassValue UML class as documented in the WaterwayClassValue section of the <a href="#">Transportation Data Dictionary</a> .

#### Class WaterwayClassValue

Definition: WaterwayClassValue is a code list used to further classify a Waterway.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.15.52. Class WaterwayFunctionValue

Requirement 258	/req/Transportation/WaterwayFunctionValue
A	Any use of the WaterwayFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterwayFunctionValue UML class as documented in the WaterwayFunctionValue section of the <a href="#">Transportation Data Dictionary</a> .

#### Class WaterwayFunctionValue

Definition: WaterwayFunctionValue is a code list that enumerates the different purposes of a Waterway.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.15.53. Class WaterwayUsageValue

Requirement 259	/req/Transportation/WaterwayUsageValue
A	Any use of the WaterwayUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterwayUsageValue UML class as documented in the WaterwayUsageValue section of the <a href="#">Transportation Data Dictionary</a> .

#### Class WaterwayUsageValue

Definition: WaterwayUsageValue is a code list that enumerates the different uses of a Waterway.

Subclass Of: <-- section,-->

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.15.54. Class GranularityValue

Requirement 260	/req/Transportation/GranularityValue
A	Any use of the GranularityValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GranularityValue UML class as documented in the GranularityValue section of the <a href="#">Transportation Data Dictionary</a> .

#### Class GranularityValue

Definition: GranularityValue enumerates the different levels of granularity in which transportation objects are represented.

Subclass Of: <-- section,-->

Stereotype:

#### Relations

#### Attributes

Attribute Name: lane

Value Type:

Definition: Indicates that the individual lanes of the transportation object are represented.

Multiplicity:

Stereotype:

Attribute Name: way

Value Type:

Definition: Indicates that the individual (carriage)ways of the transportation object are represented.

Multiplicity:

Stereotype:

### 11.15.55. Class TrafficDirectionValue

Requirement 261	/req/Transportation/TrafficDirectionValue
A	Any use of the TrafficDirectionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficDirectionValue UML class as documented in the TrafficDirectionValue section of the <a href="#">Transportation Data Dictionary</a> .

#### Class TrafficDirectionValue

Definition: TrafficDirectionValue enumerates the allowed directions of travel of a mobile object.

Subclass Of: <-- section,>>

Stereotype:

#### Relations

#### Attributes

Attribute Name: forwards

Value Type:

Definition: Indicates that traffic flows in the direction of the corresponding linear geometry.

Multiplicity:

Stereotype:

Attribute Name: backwards

Value Type:

Definition: Indicates that traffic flows in the opposite direction of the corresponding linear geometry.

Multiplicity:

Stereotype:

Attribute Name: both

Value Type:

Definition: Indicates that traffic flows in both directions.

Multiplicity:

Stereotype:

### 11.15.56. Additional Information

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

## 11.16. Tunnel

Requirements Class	
<a href="http://www.opengis.net/spec/CityGML/3.1/req/req-class-tunnel">http://www.opengis.net/spec/CityGML/3.1/req/req-class-tunnel</a>	
Target type	Conceptual Model
Dependency	<a href="#">/req/req-class-core</a>

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](#).

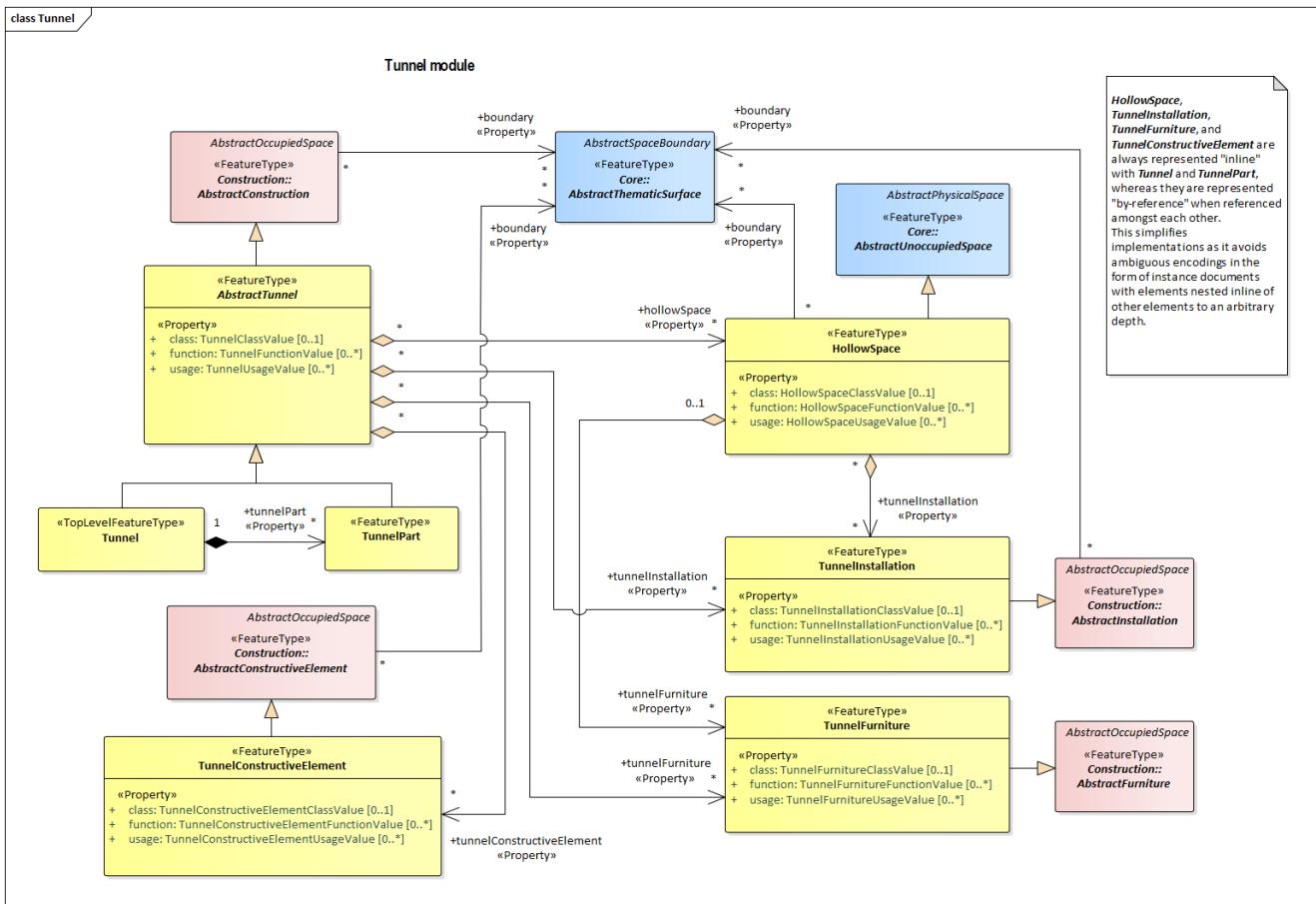


Figure 29. UML diagram of the Tunnel Model.

### 11.16.1. Tunnel Package

#### Package Tunnel

Description: The Tunnel module supports representation of thematic and spatial aspects of tunnels, tunnel parts, tunnel installations, and interior tunnel structures.

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.16.2. Class AbstractTunnel

Requirement 262	/req/Tunnel/AbstractTunnel
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the <b>AbstractTunnel</b> UML class as documented in the <a href="#">AbstractTunnel</a> section of the <a href="#">Tunnel Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the <b>AbstractTunnel</b> UML class as documented in the <a href="#">AbstractTunnel</a> section of the <a href="#">Tunnel Data Dictionary</a> .

C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractTunnel UML class; including the name, definition, type, and cardinality of those documented in the AbstractTunnel section of the <a href="#">Tunnel Data Dictionary</a> .
---	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Class AbstractTunnel

Definition: AbstractTunnel is an abstract superclass representing the common attributes and associations of the classes Tunnel and TunnelPart.

Subclass Of: [AbstractConstruction](#)

Stereotype: «FeatureType»

## Relations

Relation Name: tunnelFurniture

Cardinality: \*

Target Class: [TunnelFurniture](#)

Definition: Relates the furniture objects to the Tunnel or TunnelPart.

Relation Name: tunnelInstallation

Cardinality: \*

Target Class: [TunnelInstallation](#)

Definition: Relates the installation objects to the Tunnel or TunnelPart.

Relation Name: tunnelConstructiveElement

Cardinality: \*

Target Class: [TunnelConstructiveElement](#)

Definition: Relates the constructive elements to the Tunnel or TunnelPart.

Relation Name: hollowSpace

Cardinality: \*

Target Class: [HollowSpace](#)

Definition: Relates the hollow spaces to the Tunnel or TunnelPart.

## Attributes

Attribute Name: class

Value Type: [TunnelClassValue](#)

Definition: Indicates the specific type of the Tunnel or TunnelPart.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: [TunnelFunctionValue](#)

Definition: Specifies the intended purposes of the Tunnel or TunnelPart.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: [TunnelUsageValue](#)

Definition: Specifies the actual uses of the Tunnel or TunnelPart.

Multiplicity: [0..\*]

Stereotype: «Property»

### 11.16.3. Class HollowSpace

Requirement 263	/req/Tunnel/HollowSpace
A	The Implementation Specification SHALL contain an element with the same definition as the HollowSpace UML class as documented in the HollowSpace section of the <a href="#">Tunnel Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the HollowSpace UML class as documented in the HollowSpace section of the <a href="#">Tunnel Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the HollowSpace UML class; including the name, definition, type, and cardinality of those documented in the HollowSpace section of the <a href="#">Tunnel Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the HollowSpace UML class; including the name, definition, type, and cardinality of those documented in the HollowSpace section of the <a href="#">Tunnel Data Dictionary</a> .

#### Class 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»

#### Relations

Relation Name: boundary

Cardinality: \*

Target Class: [AbstractThematicSurface](#)

Definition:

Relation Name: tunnelFurniture

Cardinality: \*

Target Class: [TunnelFurniture](#)

Definition: Relates to the furniture objects.

Relation Name: tunnelInstallation

Cardinality: \*

Target Class: [TunnelInstallation](#)

Definition: Relates to the installation objects.

#### Attributes

Attribute Name:	class
Value Type:	HollowSpaceClassValue
Definition:	Indicates the specific type of the HollowSpace.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	HollowSpaceFunctionValue
Definition:	Specifies the intended purposes of the HollowSpace.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	HollowSpaceUsageValue
Definition:	Specifies the actual uses of the HollowSpace.
Multiplicity:	[0..*]
Stereotype:	«Property»

#### 11.16.4. Class HollowSpaceClassValue

Requirement 264	/req/Tunnel/HollowSpaceClassValue
A	Any use of the HollowSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the HollowSpaceClassValue UML class as documented in the <a href="#">Tunnel Data Dictionary</a> .

##### Class HollowSpaceClassValue

Definition: HollowSpaceClassValue is a code list used to further classify a HollowSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

##### Relations

##### Attributes

#### 11.16.5. Class HollowSpaceFunctionValue

Requirement 265	/req/Tunnel/HollowSpaceFunctionValue
A	Any use of the HollowSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the HollowSpaceFunctionValue UML class as documented in the HollowSpaceFunctionValue section of the <a href="#">Tunnel Data Dictionary</a> .

### **Class HollowSpaceFunctionValue**

Definition: HollowSpaceFunctionValue is a code list that enumerates the different purposes of a HollowSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

## **11.16.6. Class HollowSpaceUsageValue**

<b>Requirement 266</b>	<b>/req/Tunnel/HollowSpaceUsageValue</b>
A	Any use of the HollowSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the HollowSpaceUsageValue UML class as documented in the HollowSpaceUsageValue section of the <a href="#">Tunnel Data Dictionary</a> .

### **Class HollowSpaceUsageValue**

Definition: HollowSpaceUsageValue is a code list that enumerates the different uses of a HollowSpace.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

## **11.16.7. Class Tunnel**

<b>Requirement 267</b>	<b>/req/Tunnel/Tunnel</b>
A	The Implementation Specification SHALL contain an element with the same definition as the Tunnel UML class as documented in the Tunnel section of the <a href="#">Tunnel Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Tunnel UML class as documented in the Tunnel section of the <a href="#">Tunnel Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Tunnel UML class; including the name, definition, type, and cardinality of those documented in the Tunnel section of the <a href="#">Tunnel Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the Tunnel UML class; including the name, definition, type, and cardinality of those documented in the Tunnel section of the <a href="#">Tunnel Data Dictionary</a> .
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class 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»

### Relations

Relation Name: tunnelPart

Cardinality: \*

Target Class: [TunnelPart](#)

Definition: Relates the Tunnel to the tunnel parts.

### Attributes

## 11.16.8. Class TunnelClassValue

Requirement 268	/req/Tunnel/TunnelClassValue
A	Any use of the TunnelClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelClassValue UML class as documented in the TunnelClassValue section of the <a href="#">Tunnel Data Dictionary</a> .

### Class TunnelClassValue

Definition: TunnelClassValue is a code list used to further classify a Tunnel.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Relations

### Attributes

## 11.16.9. Class TunnelConstructiveElement

Requirement 269	/req/Tunnel/TunnelConstructiveElement
A	The Implementation Specification SHALL contain an element with the same definition as the TunnelConstructiveElement UML class as documented in the TunnelConstructiveElement section of the <a href="#">Tunnel Data Dictionary</a> .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TunnelConstructiveElement UML class as documented in the TunnelConstructiveElement section of the <a href="#">Tunnel Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TunnelConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the TunnelConstructiveElement section of the <a href="#">Tunnel Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TunnelConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the TunnelConstructiveElement section of the <a href="#">Tunnel Data Dictionary</a> .

### Class 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»

### Relations

### Attributes

Attribute Name: class

Value Type: TunnelConstructiveElementClassValue

Definition: Indicates the specific type of the TunnelConstructiveElement.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: TunnelConstructiveElementFunctionValue

Definition: Specifies the intended purposes of the TunnelConstructiveElement.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: TunnelConstructiveElementUsageValue

Definition: Specifies the actual uses of the TunnelConstructiveElement.

Multiplicity: [0..\*]

Stereotype: «Property»

### 11.16.10. Class TunnelConstructiveElementClassValue

Requirement 270	/req/Tunnel/TunnelConstructiveElementClassValue
-----------------	-------------------------------------------------

A	Any use of the TunnelConstructiveElementClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelConstructiveElementClassValue UML class as documented in the TunnelConstructiveElementClassValue section of the <a href="#">Tunnel Data Dictionary</a> .
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class TunnelConstructiveElementClassValue

Definition: TunnelConstructiveElementClassValue is a code list used to further classify a TunnelConstructiveElement.

Subclass Of: <-->

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.16.11. Class TunnelConstructiveElementFunctionValue

Requirement 271	/req/Tunnel/TunnelConstructiveElementFunctionValue
A	Any use of the TunnelConstructiveElementFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelConstructiveElementFunctionValue UML class as documented in the TunnelConstructiveElementFunctionValue section of the <a href="#">Tunnel Data Dictionary</a> .

### Class TunnelConstructiveElementFunctionValue

Definition: TunnelConstructiveElementFunctionValue is a code list that enumerates the different purposes of a TunnelConstructiveElement.

Subclass Of: <-->

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.16.12. Class TunnelConstructiveElementUsageValue

Requirement 272	/req/Tunnel/TunnelConstructiveElementUsageValue
A	Any use of the TunnelConstructiveElementUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelConstructiveElementUsageValue UML class as documented in the TunnelConstructiveElementUsageValue section of the <a href="#">Tunnel Data Dictionary</a> .

### **Class TunnelConstructiveElementUsageValue**

Definition: TunnelConstructiveElementUsageValue is a code list that enumerates the different uses of a TunnelConstructiveElement.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.16.13. Class TunnelFunctionValue**

<b>Requirement 273</b>	<b>/req/Tunnel/TunnelFunctionValue</b>
A	Any use of the TunnelFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelFunctionValue UML class as documented in the TunnelFunctionValue section of the <a href="#">Tunnel Data Dictionary</a> .

### **Class TunnelFunctionValue**

Definition: TunnelFunctionValue is a code list that enumerates the different purposes of a Tunnel.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

### **11.16.14. Class TunnelFurniture**

<b>Requirement 274</b>	<b>/req/Tunnel/TunnelFurniture</b>
A	The Implementation Specification SHALL contain an element with the same definition as the TunnelFurniture UML class as documented in the TunnelFurniture section of the <a href="#">Tunnel Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TunnelFurniture UML class as documented in the TunnelFurniture section of the <a href="#">Tunnel Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TunnelFurniture UML class; including the name, definition, type, and cardinality of those documented in the TunnelFurniture section of the <a href="#">Tunnel Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the TunnelFurniture UML class; including the name, definition, type, and cardinality of those documented in the TunnelFurniture section of the <a href="#">Tunnel Data Dictionary</a> .
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class 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»

### Relations

### Attributes

Attribute Name: class

Value Type: TunnelFurnitureClassValue

Definition: Indicates the specific type of the TunnelFurniture.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: TunnelFurnitureFunctionValue

Definition: Specifies the intended purposes of the TunnelFurniture.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: TunnelFurnitureUsageValue

Definition: Specifies the actual uses of the TunnelFurniture.

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.16.15. Class TunnelFurnitureClassValue

Requirement 275	/req/Tunnel/TunnelFurnitureClassValue
A	Any use of the TunnelFurnitureClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelFurnitureClassValue UML class as documented in the TunnelFurnitureClassValue section of the <a href="#">Tunnel Data Dictionary</a> .

### Class TunnelFurnitureClassValue

Definition: TunnelFurnitureClassValue is a code list used to further classify a TunnelFurniture.

Subclass Of: <-> section,>>

Stereotype: «CodeList»

### Relations

## Attributes

### 11.16.16. Class TunnelFurnitureFunctionValue

Requirement 276	/req/Tunnel/TunnelFurnitureFunctionValue
A	Any use of the TunnelFurnitureFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelFurnitureFunctionValue UML class as documented in the TunnelFurnitureFunctionValue section of the <a href="#">Tunnel Data Dictionary</a> .

#### Class TunnelFurnitureFunctionValue

Definition: TunnelFurnitureFunctionValue is a code list that enumerates the different purposes of a TunnelFurniture.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.16.17. Class TunnelFurnitureUsageValue

Requirement 277	/req/Tunnel/TunnelFurnitureUsageValue
A	Any use of the TunnelFurnitureUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelFurnitureUsageValue UML class as documented in the TunnelFurnitureUsageValue section of the <a href="#">Tunnel Data Dictionary</a> .

#### Class TunnelFurnitureUsageValue

Definition: TunnelFurnitureUsageValue is a code list that enumerates the different uses of a TunnelFurniture.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

### 11.16.18. Class TunnelInstallation

Requirement 278	/req/Tunnel/TunnelInstallation
-----------------	--------------------------------

A	The Implementation Specification SHALL contain an element with the same definition as the TunnelInstallation UML class as documented in the TunnelInstallation section of the <a href="#">Tunnel Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TunnelInstallation UML class as documented in the TunnelInstallation section of the <a href="#">Tunnel Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TunnelInstallation UML class; including the name, definition, type, and cardinality of those documented in the TunnelInstallation section of the <a href="#">Tunnel Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TunnelInstallation UML class; including the name, definition, type, and cardinality of those documented in the TunnelInstallation section of the <a href="#">Tunnel Data Dictionary</a> .

### Class 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»

### Relations

### Attributes

Attribute Name: class

Value Type: TunnelInstallationClassValue

Definition: Indicates the specific type of the TunnelInstallation.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: TunnelInstallationFunctionValue

Definition: Specifies the intended purposes of the TunnelInstallation.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: TunnelInstallationUsageValue

Definition: Specifies the actual uses of the TunnelInstallation.

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.16.19. Class TunnelInstallationClassValue

<b>Requirement 279</b>	/req/Tunnel/TunnelInstallationClassValue
A	Any use of the TunnelInstallationClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelInstallationClassValue UML class as documented in the TunnelInstallationClassValue section of the <a href="#">Tunnel Data Dictionary</a> .

### Class TunnelInstallationClassValue

Definition: TunnelInstallationClassValue is a code list used to further classify a TunnelInstallation.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.16.20. Class TunnelInstallationFunctionValue

<b>Requirement 280</b>	/req/Tunnel/TunnelInstallationFunctionValue
A	Any use of the TunnelInstallationFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelInstallationFunctionValue UML class as documented in the TunnelInstallationFunctionValue section of the <a href="#">Tunnel Data Dictionary</a> .

### Class TunnelInstallationFunctionValue

Definition: TunnelInstallationFunctionValue is a code list that enumerates the different purposes of a TunnelInstallation.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.16.21. Class TunnelInstallationUsageValue

<b>Requirement 281</b>	/req/Tunnel/TunnelInstallationUsageValue
A	Any use of the TunnelInstallationUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelInstallationUsageValue UML class as documented in the TunnelInstallationUsageValue section of the <a href="#">Tunnel Data Dictionary</a> .

### **Class TunnelInstallationUsageValue**

Definition: TunnelInstallationUsageValue is a code list that enumerates the different uses of a TunnelInstallation.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

## **11.16.22. Class TunnelPart**

Requirement 282	/req/Tunnel/TunnelPart
A	The Implementation Specification SHALL contain an element with the same definition as the TunnelPart UML class as documented in the TunnelPart section of the <a href="#">Tunnel Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TunnelPart UML class as documented in the TunnelPart section of the <a href="#">Tunnel Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the TunnelPart UML class; including the name, definition, type, and cardinality of those documented in the TunnelPart section of the <a href="#">Tunnel Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TunnelPart UML class; including the name, definition, type, and cardinality of those documented in the TunnelPart section of the <a href="#">Tunnel Data Dictionary</a> .

### **Class 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»

#### **Relations**

#### **Attributes**

## **11.16.23. Class TunnelUsageValue**

Requirement 283	/req/Tunnel/TunnelUsageValue
A	Any use of the TunnelUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelUsageValue UML class as documented in the TunnelUsageValue section of the <a href="#">Tunnel Data Dictionary</a> .

### **Class TunnelUsageValue**

Definition: TunnelUsageValue is a code list that enumerates the different uses of a Tunnel.

Subclass Of: <– section,>>

Stereotype: «CodeList»

### **Relations**

### **Attributes**

## **11.16.24. Additional Information**

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

## **11.17. Vegetation**

### **Requirements Class**

<http://www.opengis.net/spec/CityGML/3.1/req/req-class-vegetation>

Target type	Conceptual Model
Dependency	<a href="/req/req-class-core">/req/req-class-core</a>

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](#).

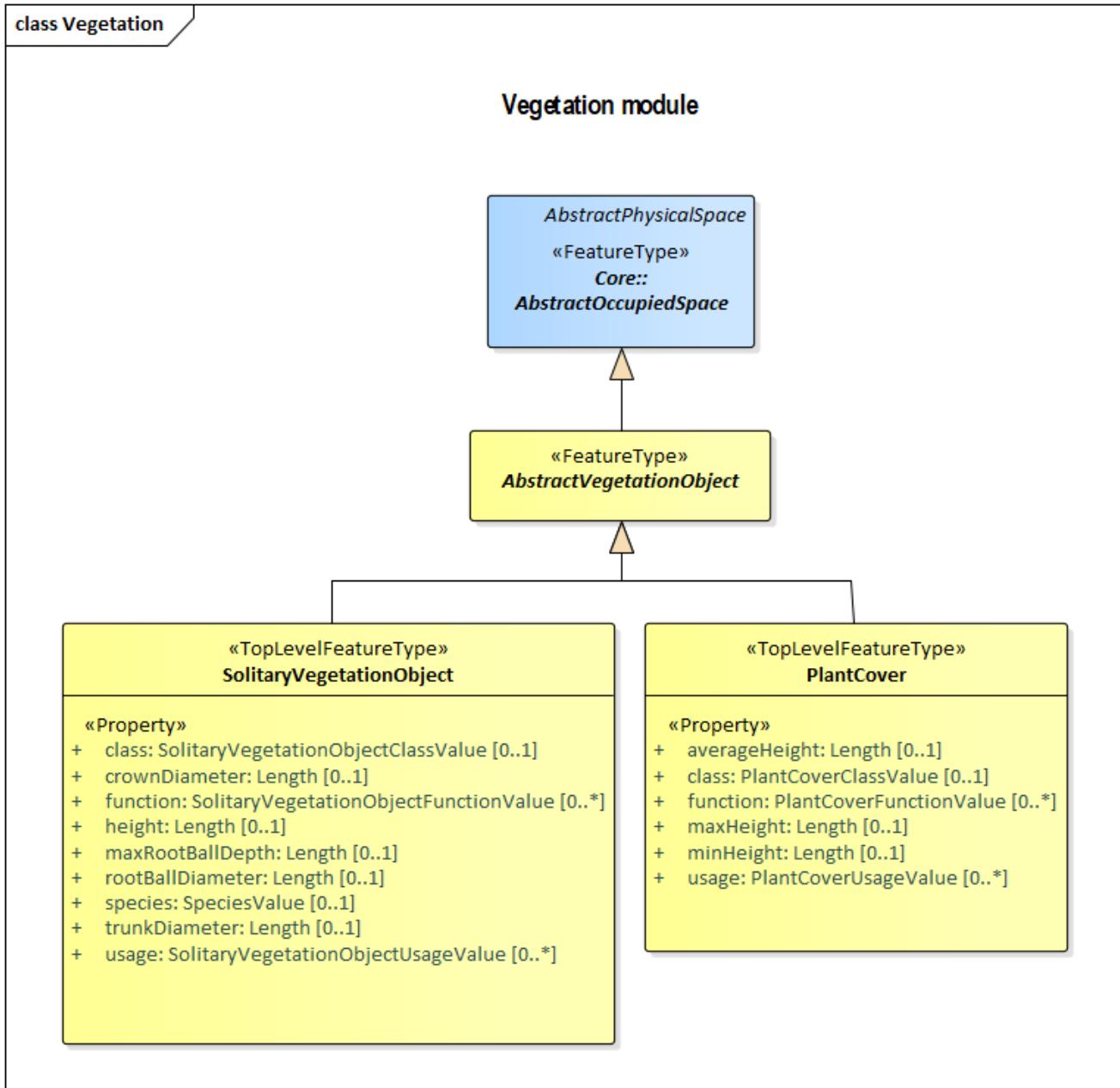


Figure 30. UML diagram of the Vegetation Model.

### 11.17.1. Vegetation 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.

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.17.2. Class AbstractVegetationObject

Requirement 284	/req/Vegetation/AbstractVegetationObject
-----------------	------------------------------------------

A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractVegetationObject UML class as documented in the AbstractVegetationObject section of the <a href="#">Vegetation Data Dictionary</a> .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractVegetationObject UML class as documented in the AbstractVegetationObject section of the <a href="#">Vegetation Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractVegetationObject UML class; including the name, definition, type, and cardinality of those documented in the AbstractVegetationObject section of the <a href="#">Vegetation Data Dictionary</a> .

### **Class AbstractVegetationObject**

Definition: AbstractVegetationObject is the abstract superclass for all kinds of vegetation objects.

Subclass Of: [AbstractOccupiedSpace](#)

Stereotype: «FeatureType»

### **Relations**

### **Attributes**

## **11.17.3. Class PlantCover**

Requirement 285	/req/Vegetation/PlantCover
A	The Implementation Specification SHALL contain an element with the same definition as the PlantCover UML class as documented in the PlantCover section of the <a href="#">Vegetation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the PlantCover UML class as documented in the PlantCover section of the <a href="#">Vegetation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the PlantCover UML class; including the name, definition, type, and cardinality of those documented in the PlantCover section of the <a href="#">Vegetation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the PlantCover UML class; including the name, definition, type, and cardinality of those documented in the PlantCover section of the <a href="#">Vegetation Data Dictionary</a> .

## Class PlantCover

Definition: A PlantCover represents a space covered by vegetation.

Subclass Of: [AbstractVegetationObject](#)

Stereotype: «TopLevelFeatureType»

## Relations

## Attributes

Attribute Name: averageHeight

Value Type: Length

Definition: Specifies the average height of the PlantCover.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: class

Value Type: PlantCoverClassValue

Definition: Indicates the specific type of the PlantCover.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: PlantCoverFunctionValue

Definition: Specifies the intended purposes of the PlantCover.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: maxHeight

Value Type: Length

Definition: Specifies the maximum height of the PlantCover.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: minHeight

Value Type: Length

Definition: Specifies the minimum height of the PlantCover.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: usage

Value Type: PlantCoverUsageValue

Definition: Specifies the actual uses of the PlantCover.

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.17.4. Class PlantCoverClassValue

Requirement 286 /req/Vegetation/PlantCoverClassValue

A	Any use of the PlantCoverClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the PlantCoverClassValue UML class as documented in the PlantCoverClassValue section of the <a href="#">Vegetation Data Dictionary</a> .
---	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class PlantCoverClassValue

Definition: PlantCoverClassValue is a code list used to further classify a PlantCover.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Relations

### Attributes

## 11.17.5. Class PlantCoverFunctionValue

Requirement 287	/req/Vegetation/PlantCoverFunctionValue
A	Any use of the PlantCoverFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the PlantCoverFunctionValue UML class as documented in the PlantCoverFunctionValue section of the <a href="#">Vegetation Data Dictionary</a> .

### Class PlantCoverFunctionValue

Definition: PlantCoverFunctionValue is a code list that enumerates the different purposes of a PlantCover.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Relations

### Attributes

## 11.17.6. Class PlantCoverUsageValue

Requirement 288	/req/Vegetation/PlantCoverUsageValue
A	Any use of the PlantCoverUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the PlantCoverUsageValue UML class as documented in the PlantCoverUsageValue section of the <a href="#">Vegetation Data Dictionary</a> .

### Class PlantCoverUsageValue

Definition: PlantCoverUsageValue is a code list that enumerates the different uses of a PlantCover.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Relations

## Attributes

### 11.17.7. Class SolitaryVegetationObject

Requirement 289	/req/Vegetation/SolitaryVegetationObject
A	The Implementation Specification SHALL contain an element with the same definition as the SolitaryVegetationObject UML class as documented in the SolitaryVegetationObject section of the <a href="#">Vegetation Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the SolitaryVegetationObject UML class as documented in the SolitaryVegetationObject section of the <a href="#">Vegetation Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the SolitaryVegetationObject UML class; including the name, definition, type, and cardinality of those documented in the SolitaryVegetationObject section of the <a href="#">Vegetation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the SolitaryVegetationObject UML class; including the name, definition, type, and cardinality of those documented in the SolitaryVegetationObject section of the <a href="#">Vegetation Data Dictionary</a> .

#### Class SolitaryVegetationObject

Definition: A SolitaryVegetationObject represents individual vegetation objects, e.g. trees or bushes.

Subclass Of: [AbstractVegetationObject](#)

Stereotype: «TopLevelFeatureType»

#### Relations

#### Attributes

Attribute Name: class

Value Type: SolitaryVegetationObjectClassValue

Definition: Indicates the specific type of the SolitaryVegetationObject.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: crownDiameter

Value Type: Length

Definition: Specifies the diameter of the SolitaryCityObject's crown.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	function
Value Type:	SolitaryVegetationObjectFunctionValue
Definition:	Specifies the intended purposes of the SolitaryVegetationObject.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	height
Value Type:	Length
Definition:	Distance between the highest point of the vegetation object and the lowest point of the terrain at the bottom of the object.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	maxRootBallDepth
Value Type:	Length
Definition:	Specifies the vertical distance between the lowest point of the SolitaryVegetationObject's root ball and the terrain surface.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	rootBallDiameter
Value Type:	Length
Definition:	Specifies the diameter of the SolitaryCityObject's root ball.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	species
Value Type:	SpeciesValue
Definition:	Indicates the botanical name of the SolitaryVegetationObject.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	trunkDiameter
Value Type:	Length
Definition:	Specifies the diameter of the SolitaryCityObject's trunk.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	SolitaryVegetationObjectUsageValue
Definition:	Specifies the actual uses of the SolitaryVegetationObject.
Multiplicity:	[0..*]
Stereotype:	«Property»

## 11.17.8. Class SolitaryVegetationObjectClassValue

Requirement 290	/req/Vegetation/SolitaryVegetationObjectClassValue
-----------------	----------------------------------------------------

A	Any use of the SolitaryVegetationObjectClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SolitaryVegetationObjectClassValue UML class as documented in the SolitaryVegetationObjectClassValue section of the <a href="#">Vegetation Data Dictionary</a> .
---	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Class SolitaryVegetationObjectClassValue

Definition: SolitaryVegetationObjectClassValue is a code list used to further classify a SolitaryVegetationObject.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.17.9. Class SolitaryVegetationObjectFunctionValue

Requirement 291	/req/Vegetation/SolitaryVegetationObjectFunctionValue
A	Any use of the SolitaryVegetationObjectFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SolitaryVegetationObjectFunctionValue UML class as documented in the SolitaryVegetationObjectFunctionValue section of the <a href="#">Vegetation Data Dictionary</a> .

### Class SolitaryVegetationObjectFunctionValue

Definition: SolitaryVegetationObjectFunctionValue is a code list that enumerates the different purposes of a SolitaryVegetationObject.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.17.10. Class SolitaryVegetationObjectUsageValue

Requirement 292	/req/Vegetation/SolitaryVegetationObjectUsageValue
A	Any use of the SolitaryVegetationObjectUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SolitaryVegetationObjectUsageValue UML class as documented in the SolitaryVegetationObjectUsageValue section of the <a href="#">Vegetation Data Dictionary</a> .

### **Class SolitaryVegetationObjectUsageValue**

Definition: SolitaryVegetationObjectUsageValue is a code list that enumerates the different uses of a SolitaryVegetationObject.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

## **11.17.11. Class SpeciesValue**

Requirement 293	/req/Vegetation/SpeciesValue
A	Any use of the SpeciesValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SpeciesValue UML class as documented in the SpeciesValue section of the <a href="#">Vegetation Data Dictionary</a> .

### **Class SpeciesValue**

Definition: A SpeciesValue is a code list that enumerates the species of a SolitaryVegetationObject.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### **Relations**

#### **Attributes**

## **11.17.12. Additional Information**

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

## **11.18. Versioning**

### **Requirements Class**

<http://www.opengis.net/spec/CityGML/3.0/req/req-class-versioning>

Target type	Conceptual Model
Dependency	<a href="/req/req-class-core">/req/req-class-core</a>

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](#).

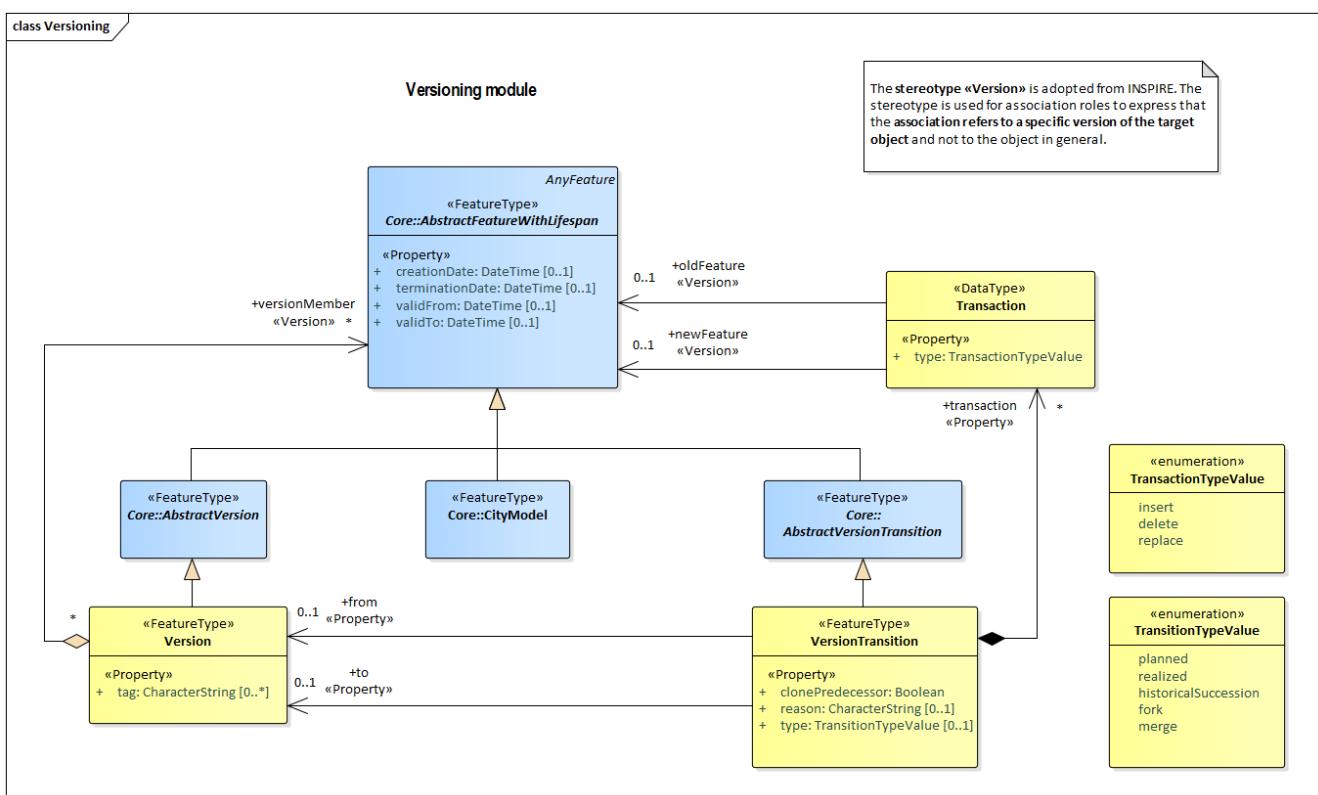


Figure 31. UML diagram of the Versioning Model.

### 11.18.1. Versioning 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.

**Stereotype:** «ApplicationSchema»

**Parent Package:** CityGML

## 11.18.2. Class Version

Requirement 294	/req/Versioning/Version
A	The Implementation Specification SHALL contain an element with the same definition as the Version UML class as documented in the Version section of the <a href="#">Versioning Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Version UML class as documented in the Version section of the <a href="#">Versioning Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Version UML class; including the name, definition, type, and cardinality of those documented in the Version section of the <a href="#">Versioning Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Version UML class; including the name, definition, type, and cardinality of those documented in the Version section of the <a href="#">Versioning Data Dictionary</a> .

### Class 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»

### Relations

Relation Name: versionMember

Cardinality: \*

Target Class: [AbstractFeatureWithLifespan](#)

Definition: Relates to all city objects that are part of the city model version.

### Attributes

Attribute Name: tag

Value Type: CharacterString

Definition: Allows for adding keywords to the city model version.

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.18.3. Class VersionTransition

Requirement 295	/req/Versioning/VersionTransition
-----------------	-----------------------------------

A	The Implementation Specification SHALL contain an element with the same definition as the VersionTransition UML class as documented in the VersionTransition section of the <a href="#">Versioning Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the VersionTransition UML class as documented in the VersionTransition section of the <a href="#">Versioning Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the VersionTransition UML class; including the name, definition, type, and cardinality of those documented in the VersionTransition section of the <a href="#">Versioning Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the VersionTransition UML class; including the name, definition, type, and cardinality of those documented in the VersionTransition section of the <a href="#">Versioning Data Dictionary</a> .

### Class 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»

### Relations

Relation Name: transaction

Cardinality: \*

Target Class: [Transaction](#)

Definition: Relates to all transactions that have been applied as part of the VersionTransition.

Relation Name: from

Cardinality: 0..1

Target Class: [Version](#)

Definition: Relates to the predecessor version of the VersionTransition.

Relation Name: to

Cardinality: 0..1

Target Class: [Version](#)

Definition: Relates to the sucessor version of the VersionTransition.

### Attributes

Attribute Name:	clonePredecessor
Value Type:	Boolean
Definition:	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).
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	reason
Value Type:	CharacterString
Definition:	Specifies why the VersionTransition has been carried out.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	type
Value Type:	TransitionTypeValue
Definition:	Indicates the specific type of the VersionTransition.
Multiplicity:	[0..1]
Stereotype:	«Property»

#### 11.18.4. Class Transaction

Requirement 296	/req/Versioning/Transaction
A	Any use of the Transaction type in the Implementation Specification SHALL have the same definition as the Transaction UML class as documented in the Transaction section of the <a href="#">Versioning Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Transaction UML class as documented in the Transaction section of the <a href="#">Versioning Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the Transaction UML class; including the name, definition, type, and cardinality of those documented in the Transaction section of the <a href="#">Versioning Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Transaction UML class; including the name, definition, type, and cardinality of those documented in the Transaction section of the <a href="#">Versioning Data Dictionary</a> .

## **Class 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»

## **Relations**

Relation Name: newFeature

Cardinality: 0..1

Target Class: [AbstractFeatureWithLifespan](#)

Definition: Relates to the version of the city object subsequent to the Transaction.

Relation Name: oldFeature

Cardinality: 0..1

Target Class: [AbstractFeatureWithLifespan](#)

Definition: Relates to the version of the city object prior to the Transaction.

## **Attributes**

Attribute Name: type

Value Type: TransactionTypeValue

Definition: Indicates the specific type of the Transaction.

Multiplicity:

Stereotype: «Property»

## **11.18.5. Class TransactionTypeValue**

Requirement 297	/req/Versioning/TransactionTypeValue
A	Any use of the TransactionTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TransactionTypeValue UML class as documented in the TransactionTypeValue section of the <a href="#">Versioning Data Dictionary</a> .

## **Class TransactionTypeValue**

Definition: TransactionTypeValue enumerates the three possible types of transactions: insert, delete, or replace.

Subclass Of: <-- section,>>

Stereotype:

## **Relations**

## **Attributes**

Attribute Name:	insert
Value Type:	
Definition:	Indicates that the feature referenced from the Transaction via the "newFeature" association has been newly created; the association "oldFeature" is empty in this case.
Multiplicity:	
Stereotype:	
Attribute Name:	delete
Value Type:	
Definition:	Indicates that the feature referenced from the Transaction via the "oldFeature" association ceases to exist; the association "newFeature" is empty in this case.
Multiplicity:	
Stereotype:	
Attribute Name:	replace
Value Type:	
Definition:	Indicates that the feature referenced from the Transaction via the "oldFeature" association has been replaced by the feature referenced via the "newFeature" association.
Multiplicity:	
Stereotype:	

## 11.18.6. Class TransitionTypeValue

Requirement 298	/req/Versioning/TransitionTypeValue
A	Any use of the TransitionTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TransitionTypeValue UML class as documented in the TransitionTypeValue section of the <a href="#">Versioning Data Dictionary</a> .

### Class 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: <– section,>>

Stereotype:

### Relations

### Attributes

Attribute Name: planned

Value Type:

Definition: Indicates that the successor version of the city model represents a planning state for a possible future of the city.

Multiplicity:

Stereotype:

Attribute Name:	realized
Value Type:	
Definition:	Indicates that the predecessor version is the chosen one from a number of possible planning versions.
Multiplicity:	
Stereotype:	
Attribute Name:	historicalSuccession
Value Type:	
Definition:	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.
Multiplicity:	
Stereotype:	
Attribute Name:	fork
Value Type:	
Definition:	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.
Multiplicity:	
Stereotype:	
Attribute Name:	merge
Value Type:	
Definition:	Indicates other reasons to converge multiple versions back into a common city model version.
Multiplicity:	
Stereotype:	

## 11.18.7. Additional Information

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

## 11.19. Water Body

Requirements Class	
<a href="http://www.opengis.net/spec/CityGML/3.1/req/req-class-waterbody">http://www.opengis.net/spec/CityGML/3.1/req/req-class-waterbody</a>	
Target type	Conceptual Model
Dependency	<a href="/req/req-class-core">/req/req-class-core</a>

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](#).

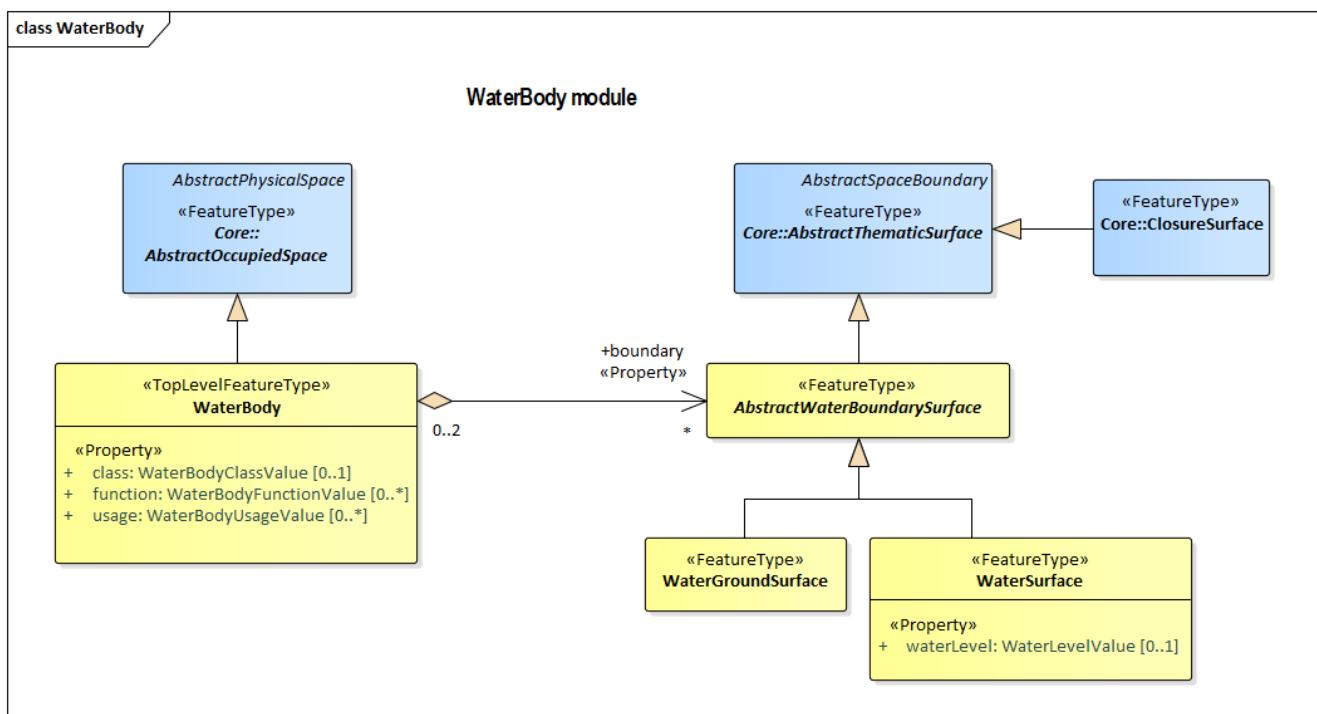


Figure 32. UML diagram of the Water Body Model.

### 11.19.1. WaterBody 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.

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.19.2. Class AbstractWaterBoundarySurface

Requirement 299	/req/Waterbody/AbstractWaterBoundarySurface
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractWaterBoundarySurface UML class as documented in the AbstractWaterBoundarySurface section of the <a href="#">Waterbody Data Dictionary</a> .

B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractWaterBoundarySurface UML class as documented in the AbstractWaterBoundarySurface section of the <a href="#">Waterbody Data Dictionary</a> .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractWaterBoundarySurface UML class; including the name, definition, type, and cardinality of those documented in the AbstractWaterBoundarySurface section of the <a href="#">Waterbody Data Dictionary</a> .

### **Class AbstractWaterBoundarySurface**

Definition: AbstractWaterBoundarySurface is the abstract superclass for all kinds of thematic surfaces bounding a water body.

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

#### **Relations**

#### **Attributes**

### **11.19.3. Class WaterBody**

Requirement 300	/req/Waterbody/WaterBody
A	The Implementation Specification SHALL contain an element with the same definition as the WaterBody UML class as documented in the WaterBody section of the <a href="#">Waterbody Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the WaterBody UML class as documented in the WaterBody section of the <a href="#">Waterbody Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the WaterBody UML class; including the name, definition, type, and cardinality of those documented in the WaterBody section of the <a href="#">Waterbody Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the WaterBody UML class; including the name, definition, type, and cardinality of those documented in the WaterBody section of the <a href="#">Waterbody Data Dictionary</a> .

## Class 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»

## Relations

Relation Name: boundary

Cardinality: \*

Target Class: [AbstractWaterBoundarySurface](#)

Definition:

## Attributes

Attribute Name: class

Value Type: WaterBodyClassValue

Definition: Indicates the specific type of the WaterBody.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: WaterBodyFunctionValue

Definition: Specifies the intended purposes of the WaterBody.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: WaterBodyUsageValue

Definition: Specifies the actual uses of the WaterBody.

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.19.4. Class WaterBodyClassValue

Requirement 301	/req/Waterbody/WaterBodyClassValue
A	Any use of the WaterBodyClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterBodyClassValue UML class as documented in the <a href="#">Waterbody Data Dictionary</a> .

## Class WaterBodyClassValue

Definition: WaterBodyClassValue is a code list used to further classify a WaterBody.

Subclass Of: <– section,>>

Stereotype: «CodeList»

## Relations

## Attributes

## 11.19.5. Class WaterBodyFunctionValue

<b>Requirement 302</b>	/req/Waterbody/WaterBodyFunctionValue
A	Any use of the WaterBodyFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterBodyFunctionValue UML class as documented in the WaterBodyFunctionValue section of the <a href="#">Waterbody Data Dictionary</a> .

### Class WaterBodyFunctionValue

Definition: WaterBodyFunctionValue is a code list that enumerates the different purposes of a WaterBody.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.19.6. Class WaterBodyUsageValue

<b>Requirement 303</b>	/req/Waterbody/WaterBodyUsageValue
A	Any use of the WaterBodyUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterBodyUsageValue UML class as documented in the WaterBodyUsageValue section of the <a href="#">Waterbody Data Dictionary</a> .

### Class WaterBodyUsageValue

Definition: WaterBodyUsageValue is a code list that enumerates the different uses of a WaterBody.

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Relations

#### Attributes

## 11.19.7. Class WaterGroundSurface

<b>Requirement 304</b>	/req/Waterbody/WaterGroundSurface
A	The Implementation Specification SHALL contain an element with the same definition as the WaterGroundSurface UML class as documented in the WaterGroundSurface section of the <a href="#">Waterbody Data Dictionary</a> .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the WaterGroundSurface UML class as documented in the WaterGroundSurface section of the <a href="#">Waterbody Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the WaterGroundSurface UML class; including the name, definition, type, and cardinality of those documented in the WaterGroundSurface section of the <a href="#">Waterbody Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the WaterGroundSurface UML class; including the name, definition, type, and cardinality of those documented in the WaterGroundSurface section of the <a href="#">Waterbody Data Dictionary</a> .

### Class WaterGroundSurface

Definition: A WaterGroundSurface represents the exterior boundary surface of the submerged bottom of a water body.

Subclass Of: [AbstractWaterBoundarySurface](#)

Stereotype: «FeatureType»

### Relations

### Attributes

## 11.19.8. Class WaterLevelValue

Requirement 305	/req/Waterbody/WaterLevelValue
A	Any use of the WaterLevelValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterLevelValue UML class as documented in the WaterLevelValue section of the <a href="#">Waterbody Data Dictionary</a> .

### Class WaterLevelValue

Definition: WaterLevelValue is a code list that enumerates the different levels of a water surface.

Subclass Of: <-> section

Stereotype: «CodeList»

### Relations

### Attributes

## 11.19.9. Class WaterSurface

Requirement 306	/req/Waterbody/WaterSurface
-----------------	-----------------------------

A	The Implementation Specification SHALL contain an element with the same definition as the WaterSurface UML class as documented in the WaterSurface section of the <a href="#">Waterbody Data Dictionary</a> .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the WaterSurface UML class as documented in the WaterSurface section of the <a href="#">Waterbody Data Dictionary</a> .
C	The implementation Specification SHALL represent the attributes of the WaterSurface UML class; including the name, definition, type, and cardinality of those documented in the WaterSurface section of the <a href="#">Waterbody Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the WaterSurface UML class; including the name, definition, type, and cardinality of those documented in the WaterSurface section of the <a href="#">Waterbody Data Dictionary</a> .

### Class WaterSurface

Definition: A WaterSurface represents the upper exterior interface between a water body and the atmosphere.

Subclass Of: [AbstractWaterBoundarySurface](#)

Stereotype: «FeatureType»

### Relations

### Attributes

Attribute Name: waterLevel

Value Type: WaterLevelValue

Definition: Specifies the level of the WaterSurface.

Multiplicity: [0..1]

Stereotype: «Property»

## 11.19.10. Additional Information

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

# **Chapter 12. 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)

## A.1. Conformance Class Appearance

<b>Abstract Test 1</b>	/ats/Appearance/AbstractSurfaceData
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-AbstractSurfaceData</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractSurfaceData abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractSurfaceData abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractSurfaceData abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 2</b>	/ats/Appearance/AbstractTexture
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-AbstractTexture</a>
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the AbstractTexture abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractTexture abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractTexture abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 3</b>	<a href="#"><b>/ats/Appearance/AbstractTextureParameterization</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Appearance/rc-AbstractTextureParameterization</b></a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractTextureParameterization abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractTextureParameterization abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractTextureParameterization abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 4</b>	<a href="#"><b>/ats/Appearance/Appearance</b></a>
------------------------	---------------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-Appearance</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Appearance class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Appearance class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Appearance class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Appearance class as documented in the Conceptual Model.

<b>Abstract Test 5</b>	<b>/ats/Appearance/Color</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-Color</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Color type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Color type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the Color type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Color type as documented in the Conceptual Model.

<b>Abstract Test 6</b>	<a href="#"><b>/ats/Appearance/ColorPlusOpacity</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Appearance/rc-ColorPlusOpacity</b></a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the ColorPlusOpacity type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the ColorPlusOpacity type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the ColorPlusOpacity type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the ColorPlusOpacity type as documented in the Conceptual Model.

<b>Abstract Test 7</b>	<a href="#"><b>/ats/Appearance/GeoreferencedTexture</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Appearance/rc-GeoreferencedTexture</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the GeoreferencedTexture class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GeoreferencedTexture class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the GeoreferencedTexture class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the GeoreferencedTexture class as documented in the Conceptual Model.

<b>Abstract Test 8</b>	<a href="#">/ats/Appearance/ParameterizedTexture</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-ParameterizedTexture</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the ParameterizedTexture class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the ParameterizedTexture class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the ParameterizedTexture class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the ParameterizedTexture class as documented in the Conceptual Model.

<b>Abstract Test 9</b>	<a href="#"><b>/ats/Appearance/TexCoordGen</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Appearance/rc-TexCoordGen</b></a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TexCoordGen type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TexCoordGen type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TexCoordGen type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TexCoordGen type as documented in the Conceptual Model.

<b>Abstract Test 10</b>	<a href="#"><b>/ats/Appearance/TexCoordList</b></a>
-------------------------	-----------------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-TexCoordList</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TexCoordList type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TexCoordList type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TexCoordList type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TexCoordList type as documented in the Conceptual Model.

<b>Abstract Test 11</b>	<a href="#">/ats/Appearance/TextureAssociation</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-TextureAssociation</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TextureAssociation class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TextureAssociation class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the TextureAssociation class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TextureAssociation class as documented in the Conceptual Model.

<b>Abstract Test 12</b>	<b>/ats/Appearance/TextureType</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-TextureType</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TextureType type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TextureType type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TextureType type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TextureType type as documented in the Conceptual Model.

<b>Abstract Test 13</b>	<b>/ats/Appearance/WrapMode</b>
-------------------------	---------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-WrapMode</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the WrapMode type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the WrapMode type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the WrapMode type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the WrapMode type as documented in the Conceptual Model.

<b>Abstract Test 14</b>	<a href="#">/ats/Appearance/X3DMaterial</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-X3DMaterial</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the X3DMaterial class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the X3DMaterial class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the X3DMaterial class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the X3DMaterial class as documented in the Conceptual Model.

## A.2. Conformance Class Bridge

<b>Abstract Test 15</b>	<a href="#">/ats/Bridge/AbstractBridge</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-AbstractBridge</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractBridge abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractBridge abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractBridge abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 16</b>	<a href="#">/ats/Bridge/Bridge</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-Bridge</a>

Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Bridge class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Bridge class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Bridge class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Bridge class as documented in the Conceptual Model.

<b>Abstract Test 17</b>	<a href="#">/ats/Bridge/BridgeClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BridgeClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 18</b>	<a href="#">/ats/Bridge/BridgeConstructiveElement</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeConstructiveElement</a>
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the BridgeConstructiveElement class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BridgeConstructiveElement class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BridgeConstructiveElement class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BridgeConstructiveElement class as documented in the Conceptual Model.

<b>Abstract Test 19</b>	<a href="#">/ats/Bridge/BridgeConstructiveElementClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeConstructiveElementClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BridgeConstructiveElementClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 20</b>	<a href="#">/ats/Bridge/BridgeConstructiveElementFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeConstructiveElementFunctionValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the BridgeConstructiveElementFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 21</b>	<a href="#"><b>/ats/Bridge/BridgeConstructiveElementUsageValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Bridge/rc-BridgeConstructiveElementUsageValue</i></a>
Test Method	Manual Inspection
A	Validate that all instances of the BridgeConstructiveElementUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 22</b>	<a href="#"><b>/ats/Bridge/BridgeFunctionValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Bridge/rc-BridgeFunctionValue</i></a>
Test Method	Manual Inspection
A	Validate that all instances of the BridgeFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 23</b>	<a href="#"><b>/ats/Bridge/BridgeFurniture</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Bridge/rc-BridgeFurniture</i></a>
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the BridgeFurniture class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BridgeFurniture class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BridgeFurniture class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BridgeFurniture class as documented in the Conceptual Model.

<b>Abstract Test 24</b>	<b>/ats/Bridge/BridgeFurnitureClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeFurnitureClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BridgeFurnitureClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 25</b>	<b>/ats/Bridge/BridgeFurnitureFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeFurnitureFunctionValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the BridgeFurnitureFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 26</b>	<a href="#">/ats/Bridge/BridgeFurnitureUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeFurnitureUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BridgeFurnitureUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 27</b>	<a href="#">/ats/Bridge/BridgeInstallation</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeInstallation</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BridgeInstallation class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BridgeInstallation class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BridgeInstallation class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the BridgeInstallation class as documented in the Conceptual Model.
---	-------------------------------------------------------------------------------------------------------------------------------------------------------------

Abstract Test 28	<b>/ats/Bridge/BridgeInstallationClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeInstallationClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BridgeInstallationClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 29	<b>/ats/Bridge/BridgeInstallationFundntionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeInstallationFundntionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BridgeInstallationFundntionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 30	<b>/ats/Bridge/BridgeInstallationUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeInstallationUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BridgeInstallationUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 31</b>	/ats/Bridge/BridgePart
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgePart</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BridgePart class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BridgePart class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BridgePart class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BridgePart class as documented in the Conceptual Model.

<b>Abstract Test 32</b>	/ats/Bridge/BridgeRoom
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeRoom</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BridgeRoom class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the BridgeRoom class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BridgeRoom class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BridgeRoom class as documented in the Conceptual Model.

<b>Abstract Test 33</b>	<b>/ats/Bridge/BridgeRoomClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeRoomClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BridgeRoomClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 34</b>	<b>/ats/Bridge/BridgeRoomFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeRoomFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BridgeRoomFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 35</b>	<a href="#">/ats/Bridge/BridgeRoomUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeRoomUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BridgeRoomUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 36</b>	<a href="#">/ats/Bridge/BridgeUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BridgeUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

### A.3. Conformance Class Building

<b>Abstract Test 37</b>	<a href="#">/ats/Building/AbstractBuilding</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-AbstractBuilding</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractBuilding abstract class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the AbstractBuilding abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractBuilding abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 38</b>	<a href="#">/ats/Building/AbstractBuildingSubdivision</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-AbstractBuildingSubdivision</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractBuildingSubdivision abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractBuildingSubdivision abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractBuildingSubdivision abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 39</b>	<a href="#">/ats/Building/Building</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Building/rc-Building</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Building class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Building class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Building class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Building class as documented in the Conceptual Model.

<b>Abstract Test 40</b>	<a href="#">/ats/Building/BuildingClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 41</b>	<a href="#">/ats/Building/BuildingConstructiveElement</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingConstructiveElement</a>

Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BuildingConstructiveElement class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BuildingConstructiveElement class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BuildingConstructiveElement class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BuildingConstructiveElement class as documented in the Conceptual Model.

<b>Abstract Test 42</b>	<a href="#">/ats/Building/BuildingConstructiveElementClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingConstructiveElementClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingConstructiveElementClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 43</b>	<a href="#">/ats/Building/BuildingConstructiveElementFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingConstructiveElementFunctionValue</a>

Test Method	Manual Inspection
A	Validate that all instances of the BuildingConstructiveElementFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 44</b>	<b>/ats/Building/BuildingConstructiveElementUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingConstructiveElementUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingConstructiveElementUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 45</b>	<b>/ats/Building/BuildingFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 46</b>	<b>/ats/Building/BuildingFurniture</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingFurniture</a>
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the BuildingFurniture class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BuildingFurniture class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BuildingFurniture class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BuildingFurniture class as documented in the Conceptual Model.

<b>Abstract Test 47</b>	<b>/ats/Building/BuildingFurnitureClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingFurnitureClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingFurnitureClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 48</b>	<b>/ats/Building/BuildingFurnitureFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingFurnitureFunctionValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the BuildingFurnitureFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 49</b>	<a href="#">/ats/Building/BuildingFurnitureUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingFurnitureUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingFurnitureUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 50</b>	<a href="#">/ats/Building/BuildingInstallation</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingInstallation</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BuildingInstallation class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BuildingInstallation class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BuildingInstallation class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the BuildingInstallation class as documented in the Conceptual Model.
---	---------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 51</b>	<b>/ats/Building/BuildingInstallationClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingInstallationClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingInstallationClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 52</b>	<b>/ats/Building/BuildingInstallationFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingInstallationFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingInstallationFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 53</b>	<b>/ats/Building/BuildingInstallationUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingInstallationUsageValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the BuildingInstallationUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 54</b>	<a href="#">/ats/Building/BuildingPart</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingPart</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BuildingPart class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BuildingPart class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BuildingPart class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BuildingPart class as documented in the Conceptual Model.

<b>Abstract Test 55</b>	<a href="#">/ats/Building/BuildingRoom</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingRoom</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BuildingRoom class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the BuildingRoom class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BuildingRoom class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BuildingRoom class as documented in the Conceptual Model.

<b>Abstract Test 56</b>	<a href="#"><b>/ats/Building/BuildingRoomClassValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Building/rc-BuildingRoomClassValue</b></a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingRoomClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 57</b>	<a href="#"><b>/ats/Building/BuildingRoomFunctionValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Building/rc-BuildingRoomFunctionValue</b></a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingRoomFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 58</b>	<a href="#">/ats/Building/BuildingRoomUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingRoomUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingRoomUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 59</b>	<a href="#">/ats/Building/BuildingSubdivisionClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingSubdivisionClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingSubdivisionClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 60</b>	<a href="#">/ats/Building/BuildingSubdivisionFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingSubdivisionFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingSubdivisionFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 61</b>	<a href="#">/ats/Building/BuildingSubdivisionUsageValue</a>
-------------------------	-------------------------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingSubdivisionUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingSubdivisionUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 62</b>	<a href="#">/ats/Building/BuildingUnit</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingUnit</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BuildingUnit class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BuildingUnit class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BuildingUnit class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BuildingUnit class as documented in the Conceptual Model.

<b>Abstract Test 63</b>	<a href="#">/ats/Building/BuildingUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	<a href="#">/req/Building/rc-BuildingUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 64</b>	<a href="#">/ats/Building/RoofTypeValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-RoofTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the RoofTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 65</b>	<a href="#">/ats/Building/RoomElevationReferenceValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-RoomElevationReferenceValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the RoomElevationReferenceValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 66</b>	<a href="#">/ats/Building/RoomHeight</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-RoomHeight</a>

Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the RoomHeight type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the RoomHeight type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the RoomHeight type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the RoomHeight type as documented in the Conceptual Model.

<b>Abstract Test 67</b>	<a href="#">/ats/Building/Storey</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-Storey</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Storey class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Storey class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Storey class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the Storey class as documented in the Conceptual Model.
---	-------------------------------------------------------------------------------------------------------------------------------------------------

## A.4. Conformance Class CityFurniture

<b>Abstract Test 68</b>	<a href="#">/ats/CityFurniture/CityFurniture</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/CityFurniture/rc-CityFurniture</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the CityFurniture class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the CityFurniture class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the CityFurniture class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the CityFurniture class as documented in the Conceptual Model.

<b>Abstract Test 69</b>	<a href="#">/ats/CityFurniture/CityFurnitureClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/CityFurniture/rc-CityFurnitureClassValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the CityFurnitureClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 70</b>	<b>/ats/CityFurniture/CityFurnitureFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/CityFurniture/rc-CityFurnitureFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the CityFurnitureFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 71</b>	<b>/ats/CityFurniture/CityFurnitureUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/CityFurniture/rc-CityFurnitureUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the CityFurnitureUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

## A.5. Conformance Class CityObjectGroup

<b>Abstract Test 72</b>	<b>/ats/CityObjectGroup/CityObjectGroup</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/CityObjectGroup/rc-CityObjectGroup</a>
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the CityObjectGroup class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the CityObjectGroup class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the CityObjectGroup class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the CityObjectGroup class as documented in the Conceptual Model.

<b>Abstract Test 73</b>	<b>/ats/CityObjectGroup/CityObjectGroupClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/CityObjectGroup/rc-CityObjectGroupClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the CityObjectGroupClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 74</b>	<b>/ats/CityObjectGroup/CityObjectGroupFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/CityObjectGroup/rc-CityObjectGroupFunctionValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the CityObjectGroupFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 75</b>	<a href="#">/ats/CityObjectGroup/CityObjectGroupUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/CityObjectGroup/rc-CityObjectGroupUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the CityObjectGroupUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 76</b>	<a href="#">/ats/CityObjectGroup/Role</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/CityObjectGroup/rc-Role</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Role class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Role class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Role class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the Role class as documented in the Conceptual Model.
---	-----------------------------------------------------------------------------------------------------------------------------------------------

## A.6. Conformance Class Core

Abstract Test 77	<a href="#">/ats/Core/AbstractAppearance</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractAppearance</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractAppearance abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractAppearance abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractAppearance abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 78	<a href="#">/ats/Core/AbstractCityObject</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractCityObject</a>
Test Method	Manual Inspection

A	Validate that a data element exists with the same definition as that of the AbstractCityObject abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractCityObject abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractCityObject abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 79</b>	<a href="#">/ats/Core/AbstractDynamizer</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractDynamizer</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractDynamizer abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractDynamizer abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractDynamizer abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 80</b>	<a href="#">/ats/Core/AbstractFeatureWithLifespan</a>
-------------------------	-------------------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractFeatureWithLifespan</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractFeatureWithLifespan abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractFeatureWithLifespan abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractFeatureWithLifespan abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 81</b>	<a href="#">/ats/Core/AbstractGenericAttribute</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractGenericAttribute</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractGenericAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractGenericAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the AbstractGenericAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the AbstractGenericAttribute type as documented in the Conceptual Model.

<b>Abstract Test 82</b>	<a href="#">/ats/Core/AbstractLogicalSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractLogicalSpace</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractLogicalSpace abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractLogicalSpace abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractLogicalSpace abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 83</b>	<a href="#">/ats/Core/AbstractOccupiedSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractOccupiedSpace</a>

Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractOccupiedSpace abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractOccupiedSpace abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractOccupiedSpace abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 84</b>	<a href="#">/ats/Core/AbstractPhysicalSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractPhysicalSpace</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractPhysicalSpace abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractPhysicalSpace abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractPhysicalSpace abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 85</b>	/ats/Core/AbstractPointCloud
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractPointCloud</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractPointCloud abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractPointCloud abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractPointCloud abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 86</b>	/ats/Core/AbstractSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractSpace</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractSpace abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractSpace abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the AbstractSpace abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 87</b>	<a href="#">/ats/Core/AbstractSpaceBoundary</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractSpaceBoundary</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractSpaceBoundary abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractSpaceBoundary abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractSpaceBoundary abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 88</b>	<a href="#">/ats/Core/AbstractThematicSurface</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractThematicSurface</a>
Test Method	Manual Inspection

A	Validate that a data element exists with the same definition as that of the AbstractThematicSurface abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractThematicSurface abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractThematicSurface abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 89</b>	<a href="#">/ats/Core/AbstractUnoccupiedSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractUnoccupiedSpace</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractUnoccupiedSpace abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractUnoccupiedSpace abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractUnoccupiedSpace abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 90</b>	<a href="#">/ats/Core/AbstractVersion</a>
-------------------------	-------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractVersion</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractVersion abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractVersion abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractVersion abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 91</b>	<b>/ats/Core/AbstractVersionTransition</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractVersionTransition</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractVersionTransition abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractVersionTransition abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the AbstractVersionTransition abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 92</b>	/ats/Core/Address
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-Address</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the Address class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Address class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Address class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Address class as documented in the Conceptual Model.

<b>Abstract Test 93</b>	/ats/Core/CityModel
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-CityModel</a>
Test Method	Manual Inspection

A	Validate that a data element exists with the same definition as that of the CityModel class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the CityModel class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the CityModel class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the CityModel class as documented in the Conceptual Model.

<b>Abstract Test 94</b>	<a href="#">/ats/Core/CityModelMember</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-CityModelMember</a>
Test Method	Manual Inspection
A	Validate that all instances of the CityModelMember codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 95</b>	<a href="#">/ats/Core/CityObjectRelation</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-CityObjectRelation</a>
Test Method	Manual Inspection

A	Validate that a data element exists with the same definition as that of the CityObjectRelation type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the CityObjectRelation type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the CityObjectRelation type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the CityObjectRelation type as documented in the Conceptual Model.

<b>Abstract Test 96</b>	<b>/ats/Core/ClosureSurface</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-ClosureSurface</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the ClosureSurface class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the ClosureSurface class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the ClosureSurface class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the ClosureSurface class as documented in the Conceptual Model.
---	---------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 97</b>	<b>/ats/Core/DoubleBetween0and1</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-DoubleBetween0and1</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the DoubleBetween0and1 type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the DoubleBetween0and1 type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the DoubleBetween0and1 type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the DoubleBetween0and1 type as documented in the Conceptual Model.

<b>Abstract Test 98</b>	<b>/ats/Core/DoubleBetween0and1List</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-DoubleBetween0and1List</a>
Test Method	Manual Inspection

A	Validate that a data element exists with the same definition as that of the DoubleBetween0and1List type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the DoubleBetween0and1List type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the DoubleBetween0and1List type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the DoubleBetween0and1List type as documented in the Conceptual Model.

<b>Abstract Test 99</b>	<a href="#">/ats/Core/DoubleList</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-DoubleList</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the DoubleList type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the DoubleList type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the DoubleList type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the DoubleList type as documented in the Conceptual Model.
---	----------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 100</b>	<a href="#">/ats/Core/ExternalReference</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-ExternalReference</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the ExternalReference type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the ExternalReference type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the ExternalReference type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the ExternalReference type as documented in the Conceptual Model.

<b>Abstract Test 101</b>	<a href="#">/ats/Core/ImplicitGeometry</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-ImplicitGeometry</a>
Test Method	Manual Inspection

A	Validate that a data element exists with the same definition as that of the ImplicitGeometry class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the ImplicitGeometry class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the ImplicitGeometry class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the ImplicitGeometry class as documented in the Conceptual Model.

<b>Abstract Test 102</b>	/ats/Core/IntegerBetween0and3
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-IntegerBetween0and3
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the IntegerBetween0and3 type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the IntegerBetween0and3 type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the IntegerBetween0and3 type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the IntegerBetween0and3 type as documented in the Conceptual Model.
---	-------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 103</b>	<a href="#">/ats/Core/IntervalValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-IntervalValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the IntervalValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 104</b>	<a href="#">/ats/Core/MimeValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-MimeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the MimeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 105</b>	<a href="#">/ats/Core/MimeTypeValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-MimeTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the MimeTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 106</b>	<a href="#">/ats/Core/Occupancy</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-Occupancy</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the Occupancy type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Occupancy type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Occupancy type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Occupancy type as documented in the Conceptual Model.

<b>Abstract Test 107</b>	<a href="#">/ats/Core/OccupantTypeValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-OccupantTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the OccupantTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 108</b>	<a href="#">/ats/Core/OtherRelationTypeValue</a>
--------------------------	--------------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-OtherRelationTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the OtherRelationTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 109</b>	<b>/ats/Core/QualifiedArea</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-QualifiedArea</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the QualifiedArea type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the QualifiedArea type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the QualifiedArea type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the QualifiedArea type as documented in the Conceptual Model.

<b>Abstract Test 110</b>	<b>/ats/Core/QualifiedAreaValue</b>
--------------------------	-------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-QualifiedAreaValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the QualifiedAreaValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 111</b>	<b>/ats/Core/QualifiedAreaTypeValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-QualifiedAreaTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the QualifiedAreaTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 112</b>	<b>/ats/Core/QualifiedVolume</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-QualifiedVolume</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the QualifiedVolume type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the QualifiedVolume type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the QualifiedVolume type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the QualifiedVolume type as documented in the Conceptual Model.

<b>Abstract Test 113</b>	<b>/ats/Core/QualifiedVolumeValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-QualifiedVolumeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the QualifiedVolumeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 114</b>	<b>/ats/Core/QualifiedVolumeTypeValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-QualifiedVolumeTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the QualifiedVolumeTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 115</b>	<b>/ats/Core/RelationTypeValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-RelationTypeValue</a>

Test Method	Manual Inspection
A	Validate that all instances of the RelationTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 116</b>	<a href="#"><b>/ats/Core/RelativeToTerrain</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Core/rc-RelativeToTerrain</i></a>
Test Method	Manual Inspection
A	Validate that all instances of the RelativeToTerrain codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 117</b>	<a href="#"><b>/ats/Core/RelativeToWater</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Core/rc-RelativeToWater</i></a>
Test Method	Manual Inspection
A	Validate that all instances of the RelativeToWater codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 118</b>	<a href="#"><b>/ats/Core/SpaceType</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Core/rc-SpaceType</i></a>
Test Method	Manual Inspection

A	Validate that all instances of the SpaceType codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 119</b>	<a href="#">/ats/Core/TemporalRelationTypeValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-TemporalRelationTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TemporalRelationTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 120</b>	<a href="#">/ats/Core/TopologicalRelationTypeValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-TopologicalRelationTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TopologicalRelationTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 121</b>	<a href="#">/ats/Core/TransformationMatrix2x2</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-TransformationMatrix2x2</a>
Test Method	Manual Inspection

A	Validate that a data element exists with the same definition as that of the TransformationMatrix2x2 type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TransformationMatrix2x2 type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TransformationMatrix2x2 type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TransformationMatrix2x2 type as documented in the Conceptual Model.

<b>Abstract Test 122</b>	<a href="#">/ats/Core/TransformationMatrix3x4</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-TransformationMatrix3x4</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the TransformationMatrix3x4 type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TransformationMatrix3x4 type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the TransformationMatrix3x4 type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TransformationMatrix3x4 type as documented in the Conceptual Model.

<b>Abstract Test 123</b>	<a href="#">/ats/Core/TransformationMatrix4x4</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-TransformationMatrix4x4</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the TransformationMatrix4x4 type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TransformationMatrix4x4 type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TransformationMatrix4x4 type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TransformationMatrix4x4 type as documented in the Conceptual Model.

<b>Abstract Test 124</b>	<a href="#">/ats/Core/XALAddressDetails</a>
--------------------------	---------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-XALAddressDetails</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the XALAddressDetails type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the XALAddressDetails type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the XALAddressDetails type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the XALAddressDetails type as documented in the Conceptual Model.

## A.7. Conformance Class Dynamizer

Abstract Test 125	<a href="#">/ats/Dynamizer/AbstractAtomicTimeSeries</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-AbstractAtomicTimeSeries</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractAtomicTimeSeries abstract class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the AbstractAtomicTimeSeries abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractAtomicTimeSeries abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 126</b>	<a href="#">/ats/Dynamizer/AbstractTimeSeries</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-AbstractTimeSeries</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the AbstractTimeSeries abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractTimeSeries abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractTimeSeries abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 127</b>	<a href="#">/ats/Dynamizer/AuthenticationValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	<a href="#">/req/Dynamizer/rc-AuthenticationValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the AuthenticationValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 128</b>	<a href="#">/ats/Dynamizer/AuthenticationTypeValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-AuthenticationTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the AuthenticationTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 129</b>	<a href="#">/ats/Dynamizer/CompositeTimeseries</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-CompositeTimeseries</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the CompositeTimeseries class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the CompositeTimeseries class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the CompositeTimeseries class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the CompositeTimeseries class as documented in the Conceptual Model.

<b>Abstract Test 130</b>	<a href="#">/ats/Dynamizer/Dynamizer</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-Dynamizer</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the Dynamizer class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Dynamizer class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Dynamizer class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Dynamizer class as documented in the Conceptual Model.

<b>Abstract Test 131</b>	<a href="#">/ats/Dynamizer/GenericTimeseries</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Dynamizer/rc-GenericTimeseries</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the GenericTimeseries class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GenericTimeseries class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the GenericTimeseries class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the GenericTimeseries class as documented in the Conceptual Model.

<b>Abstract Test 132</b>	<a href="#">/ats/Dynamizer/SensorConnection</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-SensorConnection</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the SensorConnection type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the SensorConnection type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the SensorConnection type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the SensorConnection type as documented in the Conceptual Model.

<b>Abstract Test 133</b>	<b>/ats/Dynamizer/SensorConnectionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-SensorConnectionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the SensorConnectionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 134</b>	<b>/ats/Dynamizer/SensorConnectionTypeValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-SensorConnectionTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the SensorConnectionTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 135</b>	<b>/ats/Dynamizer/StandardFileTimeseries</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-StandardFileTimeseries</a>

Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the StandardFileTimeseries class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the StandardFileTimeseries class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the StandardFileTimeseries class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the StandardFileTimeseries class as documented in the Conceptual Model.

<b>Abstract Test 136</b>	<b>/ats/Dynamizer/StandardFieldValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-StandardFieldValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the StandardFieldValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 137</b>	<b>/ats/Dynamizer/StandardFileTypeValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-StandardFileTypeValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the StandardFileTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 138</b>	<b>/ats/Dynamizer/TabulatedFileTimeseries</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-TabulatedFileTimeseries</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the TabulatedFileTimeseries class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TabulatedFileTimeseries class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TabulatedFileTimeseries class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TabulatedFileTimeseries class as documented in the Conceptual Model.

<b>Abstract Test 139</b>	<b>/ats/Dynamizer/TabulatedFileValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-TabulatedFileValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the TabulatedFileValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 140</b>	<a href="#">/ats/Dynamizer/TabulatedFileTypeValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-TabulatedFileTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TabulatedFileTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 141</b>	<a href="#">/ats/Dynamizer/TimeseriesComponent</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-TimeseriesComponent</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the TimeseriesComponent type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TimeseriesComponent type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TimeseriesComponent type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the TimeseriesComponent type as documented in the Conceptual Model.
---	-------------------------------------------------------------------------------------------------------------------------------------------------------------

Abstract Test 142	<a href="#">/ats/Dynamizer/TimeseriesValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-TimeseriesValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TimeseriesValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 143	<a href="#">/ats/Dynamizer/TimeseriesTypeValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-TimeseriesTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TimeseriesTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 144	<a href="#">/ats/Dynamizer/TimeValuePair</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-TimeValuePair</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the TimeValuePair type in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the TimeValuePair type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TimeValuePair type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TimeValuePair type as documented in the Conceptual Model.

## A.8. Conformance Class Generics

<b>Abstract Test 145</b>	/ats/Generics/DateAttribute
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-DateAttribute</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the DateAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the DateAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the DateAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the DateAttribute type as documented in the Conceptual Model.
---	-------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 146</b>	<a href="#">/ats/Generics/DoubleAttribute</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-DoubleAttribute</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the DoubleAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the DoubleAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the DoubleAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the DoubleAttribute type as documented in the Conceptual Model.

<b>Abstract Test 147</b>	<a href="#">/ats/Generics/GenericAttributeSet</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericAttributeSet</a>
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the GenericAttributeSet type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GenericAttributeSet type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the GenericAttributeSet type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the GenericAttributeSet type as documented in the Conceptual Model.

<b>Abstract Test 148</b>	<a href="#">/ats/Generics/GenericLogicalSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericLogicalSpace</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the GenericLogicalSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GenericLogicalSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the GenericLogicalSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the GenericLogicalSpace class as documented in the Conceptual Model.
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 149</b>	<a href="#">/ats/Generics/GenericLogicalSpaceClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericLogicalSpaceClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the GenericLogicalSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 150</b>	<a href="#">/ats/Generics/GenericLogicalSpaceFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericLogicalSpaceFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the GenericLogicalSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 151</b>	<a href="#">/ats/Generics/GenericLogicalSpaceUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericLogicalSpaceUsageValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the GenericLogicalSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 152</b>	<a href="#">/ats/Generics/GenericOccupiedSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericOccupiedSpace</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the GenericOccupiedSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GenericOccupiedSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the GenericOccupiedSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the GenericOccupiedSpace class as documented in the Conceptual Model.

<b>Abstract Test 153</b>	<a href="#">/ats/Generics/GenericOccupiedSpaceClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericOccupiedSpaceClassValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the GenericOccupiedSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 154</b>	<a href="#">/ats/Generics/GenericOccupiedSpaceFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericOccupiedSpaceFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the GenericOccupiedSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 155</b>	<a href="#">/ats/Generics/GenericOccupiedSpaceUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericOccupiedSpaceUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the GenericOccupiedSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 156</b>	<a href="#">/ats/Generics/GenericThematicSurface</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericThematicSurface</a>
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the GenericThematicSurface class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GenericThematicSurface class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the GenericThematicSurface class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the GenericThematicSurface class as documented in the Conceptual Model.

<b>Abstract Test 157</b>	<a href="#">/ats/Generics/GenericThematicSurfaceClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericThematicSurfaceClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the GenericThematicSurfaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 158</b>	<a href="#">/ats/Generics/GenericThematicSurfaceFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericThematicSurfaceFunctionValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the GenericThematicSurfaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 159</b>	<a href="#">/ats/Generics/GenericThematicSurfaceUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericThematicSurfaceUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the GenericThematicSurfaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 160</b>	<a href="#">/ats/Generics/GenericUnoccupiedSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericUnoccupiedSpace</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the GenericUnoccupiedSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GenericUnoccupiedSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the GenericUnoccupiedSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the GenericUnoccupiedSpace class as documented in the Conceptual Model.
---	-----------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 161</b>	<a href="#">/ats/Generics/GenericUnoccupiedSpaceClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericUnoccupiedSpaceClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the GenericUnoccupiedSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 162</b>	<a href="#">/ats/Generics/GenericUnoccupiedSpaceFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericUnoccupiedSpaceFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the GenericUnoccupiedSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 163</b>	<a href="#">/ats/Generics/GenericUnoccupiedSpaceUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericUnoccupiedSpaceUsageValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the GenericUnoccupiedSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 164</b>	<a href="#">/ats/Generics/IntAttribute</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-IntAttribute</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the IntAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the IntAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the IntAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the IntAttribute type as documented in the Conceptual Model.

<b>Abstract Test 165</b>	<a href="#">/ats/Generics/MeasureAttribute</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-MeasureAttribute</a>
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the MeasureAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the MeasureAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the MeasureAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the MeasureAttribute type as documented in the Conceptual Model.

<b>Abstract Test 166</b>	/ats/Generics/StringAttribute
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Generics/rc-StringAttribute
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the StringAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the StringAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the StringAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the StringAttribute type as documented in the Conceptual Model.
---	---------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 167</b>	<a href="#">/ats/Generics/UriAttribute</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-UriAttribute</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the UriAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the UriAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the UriAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the UriAttribute type as documented in the Conceptual Model.

## A.9. Conformance Class LandUse

<b>Abstract Test 168</b>	<a href="#">/ats/LandUse/LandUse</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/LandUse/rc-LandUse</a>
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the LandUse class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the LandUse class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the LandUse class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the LandUse class as documented in the Conceptual Model.

<b>Abstract Test 169</b>	<b>/ats/LandUse/LandUseClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/LandUse/rc-LandUseClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the LandUseClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 170</b>	<b>/ats/LandUse/LandUseFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/LandUse/rc-LandUseFunctionValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the LandUseFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 171</b>	<a href="#">/ats/LandUse/LandUseUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/LandUse/rc-LandUseUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the LandUseUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

## A.10. Conformance Class PointCloud

<b>Abstract Test 172</b>	<a href="#">/ats/PointCloud/PointCloud</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/PointCloud/rc-PointCloud</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the PointCloud class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the PointCloud class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the PointCloud class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the PointCloud class as documented in the Conceptual Model.
---	-----------------------------------------------------------------------------------------------------------------------------------------------------

## A.11. Conformance Class Relief

Abstract Test 173	<a href="#">/ats/Relief/AbstractReliefComponent</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Relief/rc-AbstractReliefComponent</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractReliefComponent abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractReliefComponent abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractReliefComponent abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 174	<a href="#">/ats/Relief/BreaklineRelief</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Relief/rc-BreaklineRelief</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BreaklineRelief class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the BreaklineRelief class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BreaklineRelief class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BreaklineRelief class as documented in the Conceptual Model.

<b>Abstract Test 175</b>	<b>/ats/Relief/MassPointRelief</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Relief/rc-MassPointRelief</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the MassPointRelief class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the MassPointRelief class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the MassPointRelief class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the MassPointRelief class as documented in the Conceptual Model.

<b>Abstract Test 176</b>	<a href="#">/ats/Relief/RasterRelief</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Relief/rc-RasterRelief</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the RasterRelief class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the RasterRelief class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the RasterRelief class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the RasterRelief class as documented in the Conceptual Model.

<b>Abstract Test 177</b>	<a href="#">/ats/Relief/ReliefFeature</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Relief/rc-ReliefFeature</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the ReliefFeature class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the ReliefFeature class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the ReliefFeature class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the ReliefFeature class as documented in the Conceptual Model.

<b>Abstract Test 178</b>	<a href="#">/ats/Relief/TINRelief</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Relief/rc-TINRelief</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TINRelief class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TINRelief class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TINRelief class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TINRelief class as documented in the Conceptual Model.

## A.12. Conformance Class Transportation

<b>Abstract Test 179</b>	<a href="#">/ats/Transportation/AbstractTransportationSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AbstractTransportationSpace</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractTransportationSpace abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractTransportationSpace abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractTransportationSpace abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 180</b>	<a href="#">/ats/Transportation/AuxiliaryTrafficArea</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficArea</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AuxiliaryTrafficArea class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the AuxiliaryTrafficArea class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AuxiliaryTrafficArea class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the AuxiliaryTrafficArea class as documented in the Conceptual Model.

<b>Abstract Test 181</b>	<a href="#">/ats/Transportation/AuxiliaryTrafficAreaClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficAreaClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the AuxiliaryTrafficAreaClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 182</b>	<a href="#">/ats/Transportation/AuxiliaryTrafficAreaFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficAreaFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the AuxiliaryTrafficAreaFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 183</b>	<a href="#">/ats/Transportation/AuxiliaryTrafficAreaUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficAreaUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the AuxiliaryTrafficAreaUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 184</b>	<a href="#">/ats/Transportation/AuxiliaryTrafficSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficSpace</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AuxiliaryTrafficSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AuxiliaryTrafficSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AuxiliaryTrafficSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the AuxiliaryTrafficSpace class as documented in the Conceptual Model.

<b>Abstract Test 185</b>	<a href="#">/ats/Transportation/AuxiliaryTrafficSpaceClassValue</a>
--------------------------	---------------------------------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficSpaceClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the AuxiliaryTrafficSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 186</b>	<b>/ats/Transportation/AuxiliaryTrafficSpaceFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficSpaceFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the AuxiliaryTrafficSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 187</b>	<b>/ats/Transportation/AuxiliaryTrafficSpaceUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficSpaceUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the AuxiliaryTrafficSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 188</b>	<b>/ats/Transportation/ClearanceSpace</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Transportation/rc-ClearanceSpace</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the ClearanceSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the ClearanceSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the ClearanceSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the ClearanceSpace class as documented in the Conceptual Model.

<b>Abstract Test 189</b>	<a href="#">/ats/Transportation/ClearanceSpaceClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-ClearanceSpaceClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the ClearanceSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 190</b>	<a href="#">/ats/Transportation/GranularityValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-GranularityValue</a>

Test Method	Manual Inspection
A	Validate that all instances of the GranularityValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 191</b>	/ats/Transportation/Hole
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Hole</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Hole class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Hole class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Hole class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Hole class as documented in the Conceptual Model.

<b>Abstract Test 192</b>	/ats/Transportation/HoleClassName
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-HoleClassName</a>
Test Method	Manual Inspection

A	Validate that all instances of the HoleClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 193</b>	<a href="#">/ats/Transportation/HoleSurface</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-HoleSurface</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the HoleSurface class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the HoleSurface class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the HoleSurface class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the HoleSurface class as documented in the Conceptual Model.

<b>Abstract Test 194</b>	<a href="#">/ats/Transportation/Intersection</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Intersection</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Intersection class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the Intersection class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Intersection class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Intersection class as documented in the Conceptual Model.

<b>Abstract Test 195</b>	<a href="#">/ats/Transportation/IntersectionClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-IntersectionClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the IntersectionClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 196</b>	<a href="#">/ats/Transportation/Marking</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Marking</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Marking class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the Marking class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Marking class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Marking class as documented in the Conceptual Model.

<b>Abstract Test 197</b>	<a href="#">/ats/Transportation/MarkingClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-MarkingClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the MarkingClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 198</b>	<a href="#">/ats/Transportation/Railway</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Railway</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Railway class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the Railway class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Railway class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Railway class as documented in the Conceptual Model.

<b>Abstract Test 199</b>	<a href="#">/ats/Transportation/RailwayClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-RailwayClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the RailwayClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 200</b>	<a href="#">/ats/Transportation/RailwayFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-RailwayFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the RailwayFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 201</b>	<a href="#">/ats/Transportation/RailwayUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-RailwayUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the RailwayUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 202</b>	<a href="#">/ats/Transportation/Road</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Road</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Road class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Road class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Road class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Road class as documented in the Conceptual Model.

<b>Abstract Test 203</b>	<a href="#">/ats/Transportation/RoadClassValue</a>
--------------------------	----------------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-RoadClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the RoadClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 204</b>	<b>/ats/Transportation/RoadFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-RoadFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the RoadFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 205</b>	<b>/ats/Transportation/RoadUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-RoadUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the RoadUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 206</b>	<b>/ats/Transportation/Section</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Transportation/rc-Section</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Section class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Section class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Section class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Section class as documented in the Conceptual Model.

<b>Abstract Test 207</b>	<b>/ats/Transportation/SectionClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-SectionClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the SectionClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 208</b>	<b>/ats/Transportation/Square</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Square</a>

Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Square class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Square class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Square class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Square class as documented in the Conceptual Model.

<b>Abstract Test 209</b>	<a href="#">/ats/Transportation/SquareClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-SquareClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the SquareClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 210</b>	<a href="#">/ats/Transportation/SquareFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-SquareFunctionValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the SquareFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 211</b>	<a href="#">/ats/Transportation/SquareUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-SquareUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the SquareUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 212</b>	<a href="#">/ats/Transportation/SurfaceMaterialValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-SurfaceMaterialValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the SurfaceMaterialValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 213</b>	<a href="#">/ats/Transportation/Track</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Track</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Track class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the Track class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Track class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Track class as documented in the Conceptual Model.

<b>Abstract Test 214</b>	<b>/ats/Transportation/TrackClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrackClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TrackClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 215</b>	<b>/ats/Transportation/TrackFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrackFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TrackFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 216</b>	<a href="#">/ats/Transportation/TrackUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrackUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TrackUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 217</b>	<a href="#">/ats/Transportation/TrafficArea</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrafficArea</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TrafficArea class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TrafficArea class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TrafficArea class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TrafficArea class as documented in the Conceptual Model.

<b>Abstract Test 218</b>	<a href="#">/ats/Transportation/TrafficAreaClassValue</a>
--------------------------	-----------------------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrafficAreaClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TrafficAreaClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 219</b>	<b>/ats/Transportation/TrafficAreaFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrafficAreaFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TrafficAreaFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 220</b>	<b>/ats/Transportation/TrafficAreaUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrafficAreaUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TrafficAreaUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 221</b>	<b>/ats/Transportation/TrafficDirectionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	<a href="#">/req/Transportation/rc-TrafficDirectionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TrafficDirectionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 222</b>	<a href="#">/ats/Transportation/TrafficSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrafficSpace</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TrafficSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TrafficSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TrafficSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TrafficSpace class as documented in the Conceptual Model.

<b>Abstract Test 223</b>	<a href="#">/ats/Transportation/TrafficSpaceClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrafficSpaceClassValue</a>

Test Method	Manual Inspection
A	Validate that all instances of the TrafficSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 224</b>	<a href="#"><b>/ats/Transportation/TrafficSpaceFunctionValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Transportation/rc-TrafficSpaceFunctionValue</b></a>
Test Method	Manual Inspection
A	Validate that all instances of the TrafficSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 225</b>	<a href="#"><b>/ats/Transportation/TrafficSpaceUsageValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Transportation/rc-TrafficSpaceUsageValue</b></a>
Test Method	Manual Inspection
A	Validate that all instances of the TrafficSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 226</b>	<a href="#"><b>/ats/Transportation/TransportationSpaceClassValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Transportation/rc-TransportationSpaceClassValue</b></a>
Test Method	Manual Inspection

A	Validate that all instances of the TransportationSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 227</b>	<a href="#">/ats/Transportation/TransportationSpaceFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TransportationSpaceFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TransportationSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 228</b>	<a href="#">/ats/Transportation/TransportationSpaceUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TransportationSpaceUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TransportationSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 229</b>	<a href="#">/ats/Transportation/Waterway</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Waterway</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Waterway class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the Waterway class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Waterway class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Waterway class as documented in the Conceptual Model.

<b>Abstract Test 230</b>	<a href="#">/ats/Transportation/WaterwayClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-WaterwayClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the WaterwayClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 231</b>	<a href="#">/ats/Transportation/WaterwayFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-WaterwayFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the WaterwayFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 232</b>	/ats/Transportation/WaterwayUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-WaterwayUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the WaterwayUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

## A.13. Conformance Class Tunnel

<b>Abstract Test 233</b>	/ats/Tunnel/AbstractTunnel
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-AbstractTunnel</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractTunnel abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractTunnel abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractTunnel abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 234</b>	/ats/Tunnel/HollowSpace
--------------------------	-------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-HollowSpace</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the HollowSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the HollowSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the HollowSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the HollowSpace class as documented in the Conceptual Model.

<b>Abstract Test 235</b>	<b>/ats/Tunnel/HollowSpaceClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-HollowSpaceClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the HollowSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 236</b>	<b>/ats/Tunnel/HollowSpaceFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	<a href="#">/req/Tunnel/rc-HollowSpaceFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the HollowSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 237</b>	<a href="#">/ats/Tunnel/HollowSpaceUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-HollowSpaceUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the HollowSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 238</b>	<a href="#">/ats/Tunnel/Tunnel</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-Tunnel</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Tunnel class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Tunnel class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the Tunnel class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Tunnel class as documented in the Conceptual Model.

<b>Abstract Test 239</b>	<a href="#">/ats/Tunnel/TunnelClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TunnelClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 240</b>	<a href="#">/ats/Tunnel/TunnelConstructiveElement</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelConstructiveElement</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TunnelConstructiveElement class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TunnelConstructiveElement class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the TunnelConstructiveElement class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TunnelConstructiveElement class as documented in the Conceptual Model.

<b>Abstract Test 241</b>	<a href="#">/ats/Tunnel/TunnelConstructiveElementClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelConstructiveElementClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TunnelConstructiveElementClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 242</b>	<a href="#">/ats/Tunnel/TunnelConstructiveElementFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelConstructiveElementFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TunnelConstructiveElementFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 243</b>	<a href="#">/ats/Tunnel/TunnelConstructiveElementUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	<a href="#">/req/Tunnel/rc-TunnelConstructiveElementUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TunnelConstructiveElementUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 244</b>	<a href="#">/ats/Tunnel/TunnelFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TunnelFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 245</b>	<a href="#">/ats/Tunnel/TunnelFurniture</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelFurniture</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TunnelFurniture class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TunnelFurniture class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the TunnelFurniture class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TunnelFurniture class as documented in the Conceptual Model.

<b>Abstract Test 246</b>	<b>/ats/Tunnel/TunnelFurnitureClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelFurnitureClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TunnelFurnitureClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 247</b>	<b>/ats/Tunnel/TunnelFurnitureFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelFurnitureFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TunnelFurnitureFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 248</b>	<b>/ats/Tunnel/TunnelFurnitureUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelFurnitureUsageValue</a>

Test Method	Manual Inspection
A	Validate that all instances of the TunnelFurnitureUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 249</b>	<b>/ats/Tunnel/TunnelInstallation</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelInstallation</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TunnelInstallation class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TunnelInstallation class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TunnelInstallation class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TunnelInstallation class as documented in the Conceptual Model.

<b>Abstract Test 250</b>	<b>/ats/Tunnel/TunnelInstallationClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelInstallationClassValue</a>
Test Method	Manual Inspection

A	Validate that all instances of the TunnelInstallationClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 251</b>	<b>/ats/Tunnel/TunnelInstallationFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelInstallationFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TunnelInstallationFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 252</b>	<b>/ats/Tunnel/TunnelInstallationUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelInstallationUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TunnelInstallationUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 253</b>	<b>/ats/Tunnel/TunnelPart</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelPart</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TunnelPart class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the TunnelPart class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TunnelPart class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TunnelPart class as documented in the Conceptual Model.

<b>Abstract Test 254</b>	<b>/ats/Tunnel/TunnelUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TunnelUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

## A.14. Conformance Class Vegetation

<b>Abstract Test 255</b>	<b>/ats/Vegetation/AbstractVegetationObject</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-AbstractVegetationObject</a>
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the AbstractVegetationObject abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractVegetationObject abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractVegetationObject abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 256</b>	<a href="#">/ats/Vegetation/PlantCover</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-PlantCover</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the PlantCover class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the PlantCover class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the PlantCover class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the PlantCover class as documented in the Conceptual Model.

<b>Abstract Test 257</b>	<b>/ats/Vegetation/PlantCoverClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-PlantCoverClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the PlantCoverClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 258</b>	<b>/ats/Vegetation/PlantCoverFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-PlantCoverFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the PlantCoverFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 259</b>	<b>/ats/Vegetation/PlantCoverUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-PlantCoverUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the PlantCoverUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 260</b>	<b>/ats/Vegetation/SolitaryVegetationObject</b>
--------------------------	-------------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-SolitaryVegetationObject</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the SolitaryVegetationObject class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the SolitaryVegetationObject class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the SolitaryVegetationObject class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the SolitaryVegetationObject class as documented in the Conceptual Model.

<b>Abstract Test 261</b>	<b>/ats/Vegetation/SolitaryVegetationObjectClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-SolitaryVegetationObjectClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the SolitaryVegetationObjectClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 262</b>	<b>/ats/Vegetation/SolitaryVegetationObjectFunctionValue</b>
--------------------------	--------------------------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-SolitaryVegetationObjectFunctionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the SolitaryVegetationObjectFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 263</b>	<a href="#">/ats/Vegetation/SolitaryVegetationObjectUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-SolitaryVegetationObjectUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the SolitaryVegetationObjectUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 264</b>	<a href="#">/ats/Vegetation/SpeciesValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-SpeciesValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the SpeciesValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

## A.15. Conformance Class Versioning

<b>Abstract Test 265</b>	<a href="#">/ats/Versioning/Transaction</a>
--------------------------	---------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Versioning/rc-Transaction</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the Transaction type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Transaction type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Transaction type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Transaction type as documented in the Conceptual Model.

<b>Abstract Test 266</b>	<a href="#">/ats/Versioning/TransactionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Versioning/rc-TransactionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TransactionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 267</b>	<a href="#">/ats/Versioning/TransactionTypeValue</a>
--------------------------	------------------------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Versioning/rc-TransactionTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TransactionTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 268</b>	<b>/ats/Versioning/TransitionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Versioning/rc-TransitionValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TransitionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 269</b>	<b>/ats/Versioning/TransitionTypeValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Versioning/rc-TransitionTypeValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the TransitionTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 270</b>	<b>/ats/Versioning/Version</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Versioning/rc-Version</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the Version class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Version class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Version class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Version class as documented in the Conceptual Model.

<b>Abstract Test 271</b>	<a href="#">/ats/Versioning/VersionTransition</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Versioning/rc-VersionTransition</a>
Test Method	Manual Inspection
A	Validate that a data element exists with the same definition as that of the VersionTransition class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the VersionTransition class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the VersionTransition class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the VersionTransition class as documented in the Conceptual Model.

## A.16. Conformance Class WaterBody

<b>Abstract Test 272</b>	<a href="#">/ats/Waterbody/AbstractWaterBoundarySurface</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Waterbody/rc-AbstractWaterBoundarySurface</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractWaterBoundarySurface abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractWaterBoundarySurface abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractWaterBoundarySurface abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

<b>Abstract Test 273</b>	<a href="#">/ats/Waterbody/WaterBody</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Waterbody/rc-WaterBody</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the WaterBody class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the WaterBody class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the WaterBody class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the WaterBody class as documented in the Conceptual Model.

<b>Abstract Test 274</b>	<a href="#">/ats/Waterbody/WaterBodyClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Waterbody/rc-WaterBodyClassValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the WaterBodyClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 275</b>	<a href="#">/ats/Waterbody/WaterBodyFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Waterbody/rc-WaterBodyFunctionValue</a>

Test Method	Manual Inspection
A	Validate that all instances of the WaterBodyFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 276</b>	<a href="#">/ats/Waterbody/WaterBodyUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Waterbody/rc-WaterBodyUsageValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the WaterBodyUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 277</b>	<a href="#">/ats/Waterbody/WaterGroundSurface</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Waterbody/rc-WaterGroundSurface</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the WaterGroundSurface class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the WaterGroundSurface class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the WaterGroundSurface class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the WaterGroundSurface class as documented in the Conceptual Model.
---	-------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Abstract Test 278</b>	<b>/ats/Waterbody/WaterLevelValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Waterbody/rc-WaterLevelValue</a>
Test Method	Manual Inspection
A	Validate that all instances of the WaterLevelValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

<b>Abstract Test 279</b>	<b>/ats/Waterbody/WaterSurface</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Waterbody/rc-WaterSurface</a>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the WaterSurface class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the WaterSurface class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the WaterSurface class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the WaterSurface class as documented in the Conceptual Model.
---	-------------------------------------------------------------------------------------------------------------------------------------------------------

## Annex B: Revision History

Date	Release	Editor	Primary clauses modified	Description
2016-04-28	0.1	G. Editor	all	initial version

# Chapter 13. 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

*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

## NOTE

- 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).