# TOWARDS A GENERIC 3D STANDARDISATION APPROACH FOR THE NETHERLANDS SUPPORTING DIFFERENT APPLICATIONS AND ENCODINGS

J.E. Stoter[1,2,3], H. Ledoux[1], F. Penninga[2], L. van den Brink[2], M. Reuvers[3], M. Vermeij[4], M.G. Wiersma[1]

[1]3D Geoinformation group, Delft University of Technology, [j.e.stoter| h.ledoux]@tudelft.nl, M.G.Wiersma@student.tudelft.nl
[2]Geonovum, the Netherlands, [j.stoter|f.penninga | l.vandenbrink]@geonovum.nl
[3]Kadaster, the Netherlands, [jantien.stoter | marcel.reuvers]@kadaster.nl
[4]City of Rotterdam, the Netherlands, mj.vermeij@rotterdam.nl

**KEY WORDS:** 3D standardisation, 3D data models, 3D applications, UML, JSON, Linked data

**ABSTRACT:**

In this paper we present an open and flexible approach for the standardisation of 3D geographical data, describing our physical environment in such a way that it can serve different applications. The aim of our approach is to keep the standard as simple as possible so that implementation in different software is straightforward and the reuse of once collected 3D data in different domains is optimally supported. Therefore, we propose to model the semantics of real-world objects independent from their application and we distinguish between the conceptual model and encoding. The result is a 3-layer approach, in which the first layer contains the conceptual model: the object types with their definitions and properties. This layer reuses definitions of various existing models (national and international) as much as possible. The second layer contains the modelling constraints: the set of rules that define how the objects from the conceptual model are represented in 3D as needed for a specific context or application. This second layer contains additional (3D) requirements to standardise the 3D representations of the objects. The third layer contains encoding profiles, thus specifying how different formats can best be encoded; these formats could be JSON or XML/GML.

In this paper we motivate and describe our approach. For a small area we have developed a prototype that implements the 3 different layers. The prototype shows how the approach can be implemented for one specific application and additionally it provides insight into further development.

## 1. INTRODUCTION

More and more governmental organisations use 3D data for urban planning, design, and maintaining public space (Biljecki et al., 2015). With this increased use of 3D data, there is an increasing need for 3D data that serve a wide variety of urban applications. To enable consistent and efficient use of 3D data, 3D base data of our environment should be available in a standardised way, so that the data can be collected, updated and maintained once and can be re-used by many applications.

In the Netherlands, the information model for large-scale topography (IMGeo) contains support for 3D information describing our physical environment because it is modelled as an Application Domain Extension (ADE) of CityGML (see Stoter et al (2013); Brink et al (2013)). In theory this could be the countrywide 3D base date serving a wide variety of applications. But its implementation and use in practice are still very limited. The standard is being used as an information model, and once collected 3D data can be (and is) structured in a CityGML-compliant format. However, due to limited software support, users have great problems using the CityGML data in their GIS packages and update the CityGML data. As a consequence, many users and organisations are still finding their own solution (in proprietary formats) to model and maintain 3D data, which makes the reuse of once collected 3D data difficult.

The Netherlands has a rich history of standardising geospatial data that makes reuse of geospatial data within specific applications and domains possible. There are for example domain-specific information models for large-scale topography, cables and pipelines, nature, spatial planning, underground, archaeology, building and addresses (Brink et al, 2017). At this moment, these information models are mainly limited to 2D description of the concepts involved. Since more and more organisations and applications work with 3D data, there is a growing need for a uniform approach to define 3D representations of concepts that are already defined in these different domains. Therefore, the question is whether 3D aspects in the different domain models can be modelled in a uniform way and if the reuse of once collected 3D (base) data can be increased.

To address the issue of a lacking 3D standard beyond individual applications, we started an initiative to investigate the possibility of a uniform approach that does align to international standards, but also provides the flexibility to integrate 3D data from different domains (and hence from different information models).

With a few stakeholders (Geonovum, City of Rotterdam, Kadaster and 3D Geoinformation research group at TU Delft) we have made the first steps to define an approach that meets these requirements, and we will involve more stakeholders in further development of this approach.

The approach consists of three layers that will be introduced below (Section 3), after listing the principles defining and motivating the scope of the approach (Section 2). For a small test area, we have developed a prototype that implements this approach to provide us insight on how to best further develop and refine the approach and how to implement it for a specific application (Section 4). We end with conclusions and future work in Section 5.

## 2. THE SCOPE OF OUR APPROACH

There are several considerations that define the scope of our 3D standardisation approach:

**A clear distinction between conceptual model and encoding is important** Current standards on 3D, such as CityGML, operate on two levels: they contain both a conceptual model (how to describe objects) and an encoding (how to store and exchange information about objects). To prepare for the future and support the implementation possibility of the approach, a clear distinction between the conceptual model and different encodings to support different applications and uses is important (and is also one of the main changes in the next version of CityGML which is currently under revision). In addition, the encoding should support the access to individual instances of data sets instead of only supporting exchange of complete data sets.

**Uniform 3D support in different application models without developing an overall 3D standard** We do not deem it feasible to model concepts with their 3D definitions in one overarching 3D data model that would serve all use cases and applications. Such an "all purpose" 3D standard would confront each individual application with unwanted and irrelevant complexity, which in its turn would make implementation and the maintenance of datasets more complex.

**Reuse definitions of existing information models** In reality there are no specific 3D objects, only 3D representations of objects (which probably also have one or more 2D representations depending on the context and application in which the data are used). Consequently, the approach should align to existing domain standards and reuse definitions of concepts that have already been defined in domain models as much as possible.

**Define limitations and refinements to improve standardisation** Standardisation is about making choices. For example, GML has 25 different ways of representing a simple polygon (Rouault, 2014). By restricting the degrees of freedom, it becomes more predictable which form to expect and other forms do not need to be supported. We believe that, from the perspective of implementation, it is better to limit the degrees of freedom in a standard than to support all possibilities. This is in sharp contrast to CityGML. These limitations should be reflected in (application specific) modelling constraints which, for example, allows only solids as geometric representations for a building, and not multi-surfaces or point clouds when volumes are needed for spatial analyses.

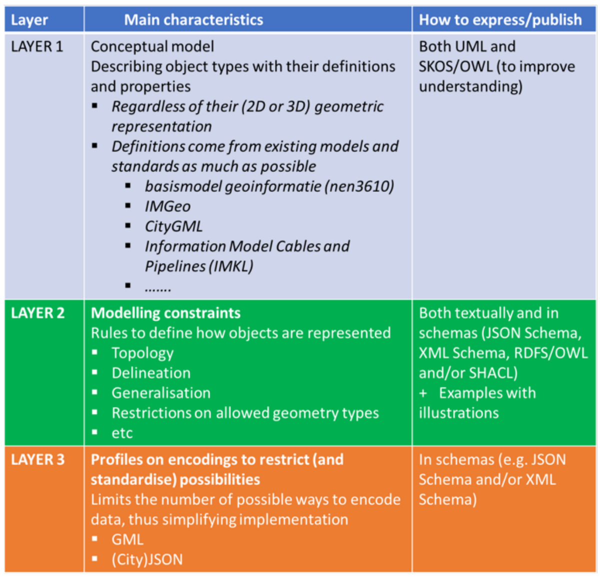
**The 3D standardisation approach should be generic** The approach should align to the national and international standards; it should support concepts that are already defined in a wide variety of information and data models and it should support several encodings to meet different purposes.

## 3. THE 3-LAYER APPROACH

This section describes the three layers of our 3D standardisation framework (Section 3.1). Section 3.2 describes the formal languages that we plan to use to express the information in each layer and Section 3.3 describes how a 3D standard for a specific application can be developed using our 3D standardisation framework.

### 3.1 Explanation of the 3-layers approach

The considerations described in Section 2 have resulted in the following 3-layer approach, see Figure 1:



| Layer | Main characteristics | How to express/publish |
|---|---|---|
| LAYER 1 | Conceptual model<br>Describing object types with their definitions and properties<br>▪ *Regardless of their (2D or 3D) geometric representation*<br>▪ *Definitions come from existing models and standards as much as possible*<br>  ▪ *basismodel geoinformatie (nen3610)*<br>  ▪ *IMGeo*<br>  ▪ *CityGML*<br>  ▪ *Information Model Cables and Pipelines (IMKL)*<br>  ▪ ……. | Both UML and SKOS/OWL (to improve understanding) |
| LAYER 2 | **Modelling constraints**<br>Rules to define how objects are represented<br>▪ Topology<br>▪ Delineation<br>▪ Generalisation<br>▪ Restrictions on allowed geometry types<br>▪ etc | Both textually and in schemas (JSON Schema, XML Schema, RDFS/OWL and/or SHACL)<br>+ Examples with illustrations |
| LAYER 3 | **Profiles on encodings to restrict (and standardise) possibilities**<br>Limits the number of possible ways to encode data, thus simplifying implementation<br>▪ GML<br>▪ (City)JSON | In schemas (e.g. JSON Schema and/or XML Schema) |

Figure 1: Schema of the proposed 3-layer approach to standardise 3D geospatial modelling in the Netherlands.

**The first layer** contains the conceptual model: the object types with their definitions and their properties. Object types are described regardless of their geometric representation. The first layer will as much as possible make use of concept-descriptions available in existing standards such as CityGML (and thus IMGeo) and extend these with definitions that are missing (such as cables and pipelines and other constructions).

We foresee two layers: the first layer (1A) is a container of all known concepts. The main source for these object type descriptions is NEN3610 (NEN, 2016). This (national) model defines concepts at a generic level. It contains concepts, definitions, relations and general rules for the exchange of location-related concepts. Based on this model, specific domains can further detail the concepts from NEN3610.

Layer 1B of the conceptual model defines the semantics for the specific applications. This is done by selecting all the generic concepts defined in the layer 1A and also by extending these definitions with application specific attributes for specific object types.

**The second layer** contains the modelling constraints within the context of domain and application: the set of rules that define how the objects from the conceptual model are represented in 3D (and in 2D for 2D applications). This layer contains additional (3D) requirements to standardise the 3D representations of the objects. This can be geometric and topological requirements but also other requirements.

These constraints, which can be 2D and 3D, can be different for different applications. It may contain certain limitations. For example, the decision to only use `Buildings` instead of `BuildingParts` because of the ambiguities that are observed in the latter (Eriksson et al., 2018). It can also restrict the allowed geometry types (such as only solids for the volumetric part of a building, as mentioned above). The constraints can also cover topological rules (e.g. `Buildings` are not allowed to overlap with each other), delineation, or generalisation rules. This layer addresses the principle of having a set of (base) objects that have a specific representation based on their context and application. The modelling constraints can be either mandatory or optional, are application-dependent and provide the validation rules of the 3D data encoded in layer 3.

**The third layer** specifies how to encode the data model and the constraints in a few specific formats. The formats can be for instance JSON and XML/GML, or others if suitable. The aim of limiting the number of possible ways to encode 3D data is to simplify the implementation for developers: the fewer possibilities that need to be supported, the simpler the implementation. And, as a consequence, the more software will potentially support the standard, and the fewer bugs there will be in those implementations.

The domain and context specific 3D data will in the end be encoded according to the specifications in layer 3 and available in these formats. If a domain model will support more than one encoding, we will pay specific attention to the interoperability of the resulting encodings.

### 3.2 Formalisation languages of the 3 layers

To improve the understanding of 3D data models, the concepts in layer 1 will not only be modelled in UML, but also published in SKOS / OWL. Layer 2, the modelling constraints, will be formulated textually, and the constraints that can be formalised will be put in schemas such as JSON Schema, XML Schema, RDFS / OWL and / or SHACL. The profiles on generic encodings in layer 3 are expressed in JSON Schema and / or XML Schema. We will base the JSON encoding on CityJSON, see Ledoux et al. (2019).

### 3.3 Developing a 3D standard for a specific use case

The approach will be embedded in the new version of the Base Model Geoinformation (NEN3610), which is currently being revised (Geonovum, 2019). This will enable us to support any use case that we would like to extend to 3D. The revised NEN3610 will then describe how, for a specific application, one can define a standard for 3D data using the 3-layer approach (and for 2D data if the specific context or application requires 2D data).

The development of a 3D information model for a specific use case (e.g. 3D data for energy consumption prediction, 3D data for flooding simulations, 3D input data for noise simulation (Kumar, 2017), etc.) comes down to answering three questions in accordance with the 3-layer approach:

1. What object types are needed for the application or domain of interest? Which object types do we need to model? Is it possible to use concepts from NEN3610 and extend these (otherwise concepts need to be added at the generic level)? Which (domain) standards do already model these concepts and can we reuse the semantics?
2. How to best and unambiguously capture these objects in 3D?
3. How do I encode the related data in a specific format?

## 4. PROTOTYPE OF THE 3-LAYER APPROACH

For a small area (see Figure 2) we have developed a prototype that implements the 3 different layers for a selection of topographical objects.
The test area contains both data that is currently supported in CityGML (version 2.0) and data for which the definitions need to be taken from other data models (e.g. a pipeline and the roots of a tree).
The aim of the prototype is firstly to show how the approach can be implemented for one specific application and secondly to provide insight into further development.

The prototype formalises the 3D data for all 3 layers in relevant languages and formats: layer 1 firstly in UML (RDW/OWL is work in progress); the layer 2 modelling guidelines are formulated in text (and later in schemas) and layer 3 uses JSON encodings (at this moment using CityGML `GenericCityObject`, but this will be modified for the specific objects taken from different standards).

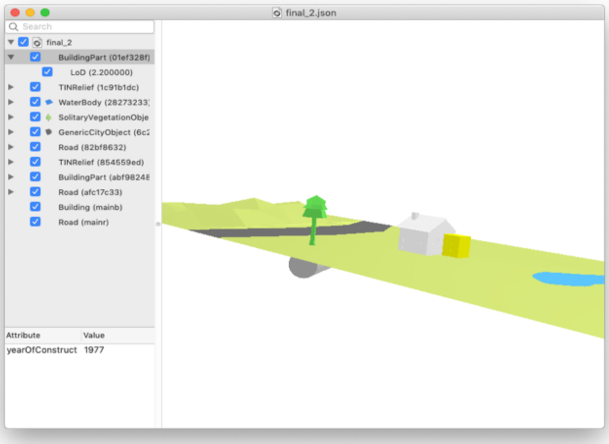


Figure 2: Test data used to develop a prototype that implements the 3-layer approach

### 4.1 Layer 1: conceptual model

For the prototype we have designed the UML diagram (Section 4.1.1) and are currently working on expressing the conceptual model in Ontology Web Language (OWL) (Section 4.1.2).

**4.1.1 Expressing the conceptual layer in UML** The UML diagram describing the concepts in our test data set are shown in Figure 3.

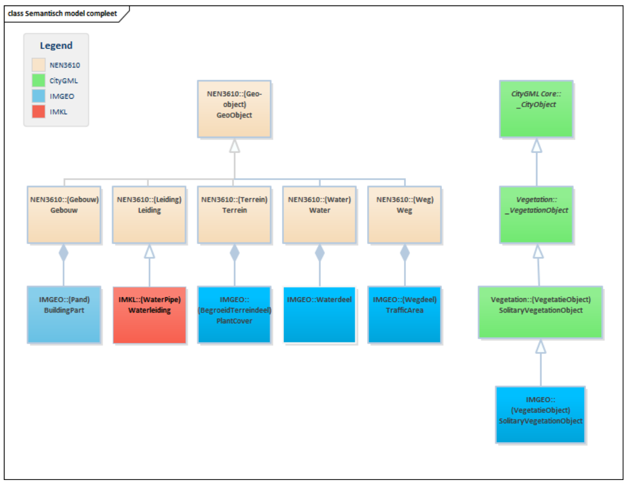

class Semantisch model compleet
Legend
NEN3610
CityGML
IMGEO
IMKL
NEN3610::(Geo-object) GeoObject
CityGML Core:: _CityObject
NEN3610::(Gebouw) Gebouw
NEN3610::(Leiding) Leiding
NEN3610::(Terrein) Terrein
NEN3610::(Water) Water
NEN3610::(Weg) Weg
Vegetation:: _VegetationObject
IMGEO::(Pand) BuildingPart
IMKL::(WaterPipe) Waterleiding
IMGEO:: (BegroeidTerreindeel) PlantCover
IMGEO::Waterdeel
IMGEO::(Wegdeel) TrafficArea
Vegetation::(VegetatieObject) SolitaryVegetationObject
IMGEO:: (VegetatieObject) SolitaryVegetationObject

Figure 3: UML diagram showing the overview of the layer 1 classes used in the prototype

Layer 1A defines concepts at general level and starts with concepts available in NEN3610: Building (Gebouw), Pipeline (Leiding), PlantCover (Terrein), Water (Water) and Road (Weg).

Because of a missing concept for solitary vegetation objects in the current NEN3610, this concept is added from CityGML. For layer 1B, concepts are refined via existing information models (as also captured in the UML diagram): Building, Plant Cover, Water, Road and Solitary Vegetation Object all reuse the definitions in IMGeo (and thus CityGML); pipeline reuses (and further refines) the definition of this concept as defined in the Dutch information model for cables and pipelines (IMKL). IMKL is an extension of the INSPIRE theme on Utilities (INSPIRE, 2013).

The `Building` concept in current IMGeo extends the class `BuildingPart` from CityGML. However, it might be better to only use `Buildings` instead of `BuildingParts` because of the ambiguities that we observed in the latter (also in relation to `BuildingUnits` as proposed in version 3.0 of CityGML). In addition, in CityGML 2.0 geometry and attributes can be assigned to both `Building` or `BuildingPart`, which is also confusing to distinguish between both types.

On the other hand, the definition of a `Building` may also give questions: it could be considered as a construction with the same year of construction. However, with this definition, the unity of one building can be broken when an extension of a building will become a different entity based on a newer year of construction.

Considering all these aspects, a better possibility may be to simply define a `Building` as a container of one or two `BuildingParts`, which can have a geometry, year of construction and other attributes. We first have to revise the definition in IMGeo before we can reconsider the definition of the building concept (and its parts) in our new approach.

**4.1.2 Expressing the conceptual layer in OWL** Currently we are also working on expressing the conceptual model in Ontology Web Language (OWL) to improve the understanding of 3D data models and to be able to publish the 3D data as linked data.

There have already been successful cases of transformations from data in GML to RDF and additional creation of OWL ontologies based on the model's UML specifications.

For our approach we convert the UML diagrams of the conceptual models of interest (CityGML 2.0, IMGeo, IMKL) into OWL (see figure 4).

```
@prefix : <http://www.example.org/ont/imgeo#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@base <http://www.example.org/ont/imgeo> .

<http://www.example.org/ont/imgeo> rdf:type owl:Ontology ;
              owl:imports <http://www.example.org/ont/bldg> ;
              owl:versionIRI "http://www.example.org/ont/imgeo/1.0" ;
              owl:versionInfo "1.0" ;
              skos:definition "Informatie Model Geografie"@nl .

#################################################################
#    Data properties
#################################################################

###  http://www.example.org/ont/imgeo#identificatieBAGPND
:identificatieBAGPND rdf:type owl:DatatypeProperty ;
             rdfs:domain <http://www.example.org/ont/bldg#BuildingPart> ;
             rdfs:range xsd:string .


###  http://www.example.org/ont/imgeo#inOnderzoek
:inOnderzoek rdf:type owl:DatatypeProperty ;
             rdfs:domain <http://www.example.org/ont/core#_CityObject> ;
             rdfs:range xsd:boolean .
```

Figure 4: Fragment with vocabulary entries involving CityGML modules and IMGeo, using Turtle syntax

An initial ontology has been created from these three data models using an automated mapping tool (i.e. ShapeChange (n.d.)). This Java tool allows ontology representations to be obtained based on input parameters and conversion rules as defined by ISO 19150-2 (ISO, 2015). However, due to the differences between UML class models and OWL – such as the use of open-world and closed-world assumptions or the notion of abstract classes – manual fixes to the resulting ontology are necessary. Finally, to test if the ontology refers to the universe of discourse modelled by the UML model, reasoner tools are used to look for logical inconsistencies.

The same logic will be applied to the encodings (see layer 3) to be able to access individual instances of data sets with their semantics via linked data technology (instead of having to exchange whole data sets).

## 4.2 Layer 2: Modelling constraints

For the modelling constraints of large scale topography we use (IMGeo, 2017) as a basis, see appendix A for a selection. These guidelines define how the CityGML standard should be used to model IMGeo-specific 3D information, and they contain modelling constraints for all object types (buildings, vegetation, tunnels bridges, etc) for all levels of detail, including what geometry types to us.

Examples of guidelines are: LOD0 geometries of all terrain polygons (water, road, building, land use, vegetation) at ground level should form a planar partition in 2.5D (no holes or overlap); water surface should be flat and horizontal; the lower surfaces of the building's block geometry must correspond to the 2D footprint as well as to the LOD0 floor surface etc.

## 4.3 Layer 3: Encoding

In the third layer, the data is at this moment encoded in CityJSON, which is a JSON-based encoding of the CityGML v2.0.0 data model. The full details of CityJSON are available in Ledoux et al. (2019).

The aim of CityJSON is to offer an alternative to the GML encoding of CityGML, which can be verbose and and thus rather complex to work with. CityJSON aims at being easy-to-use, both for reading datasets, and for creating them. It was designed with programmers in mind, so that tools and APIs supporting it can be quickly built.

There are already several open-software supporting CityJSON, and adding functionality to existing ones is simpler than if CityGML was used (because the modelling possibilities are restricted). One advantage is also that it is on average six times more compact than CityGML files, an important factor when developing an encoding for a whole country.

While CityJSON adopts the CityGML data model, it can be easily extended with new objects, in a similar fashion to CityGML ADEs. This can be done either by modifying CityJSON to make a "Dutch version" of it, or to use Extensions to model the Dutch particularities. We plan to use the latter solution since it enables the reuse of the CityJSON core by other applications/countries/organisations and to model their particularities also as Extensions.

Publishing the 3D data as linked data will enable reuse of the data beyond the use of individual data sets. Therefore, we are also investigating the enrichment of CityJSON files with machinereadable semantics. A possible option is to use JSON-LD, a serialization format for Linked Data (Sporny, Kellogg & Lanthaler, 2014). This would allow linking properties from the JSON file to the ontology concepts as defined in layer 1.

## 5. CONCLUSION AND FUTURE WORK

In this paper we presented our generic standardisation framework to standardise 3D data for different applications in a uniform and generic way. "Flexibility" and "easy to implement in software" are the main drivers for our approach in order to optimally support reuse of once collected 3D data. This addresses a challenge that will become more relevant in the near future as 3D data will be required for an increasing number of applications.

The proposed framework consists of 3 layers which are all presented. To show how the framework can be used to standardise 3D data for a specific application, we have developed a small prototype for large scale topographic data. This process of reverse engineering showed how existing 3D data can be standardised in a generic, light way supporting different applications and encodings, building on existing domain models and standards.

In a next step we will further develop our work (i.e. the expression of information in the different layers using different languages) and apply our approach to three or four real topographic data sets as needed in different applications. Linked data technologies will be incorporated for optimal access to and reuse of the involved 3D data. From those experiences we will further develop the approach into a standardisation framework for 3D data serving different applications in the Netherlands. We will also investigate how we can best define Dutch specific aspects for the international standards that we use/extend as well as how we can accommodate possible extensions for user-specific aspects.

## 6. ACKNOWLEDGEMENTS

This work has received funding from the European Research Council under the European Union's Horizon 2020 ERC Agreement no. 677312 UMnD: Urban modelling in higher dimensions.

## 7. REFERENCES

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Cöltekin, A., 2015. Applications of 3D city models: State of the art review. ISPRS International Journal of Geo-Information, 4(4):2220–9964.

Brink, van der, L., J. Stoter and S. Zlatanova, 2013. UML-Based approach to developing a CityGML Application Domain Extension. Transactions in GIS 17(6), 2013, pp. 920-942.

Brink, van der, L., P. Janssen, W. Quak and J. Stoter, 2017. Towards a high level of semantic harmonisation in the geospatial domain. Computers, Environment and Urban Systems 62, 2017, pp. 233–242.

Geonovum, 2019. https://www.geonovum.nl/geo-standaarden/nen-3610-basismodel-voor-informatiemodellen

INSPIRE, 2013. https://inspire.ec.europa.eu/id/document/tg/us

IMGeo, 2017. Technical Specifications for the Construction of 3D IMGeo-CityGML https://www.geonovum.nl/uploads/documents/20170102Guidetotender3DCityGMLIMGeo_v2.1_0.pdf

ISO, 2015. ISO 19150–2:2015 Geographic information - Ontology - Part 2, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=57466

Kumar, K., H. Ledoux, T. Commandeur, and J. Stoter, 2017. Modelling urban noise in CityGML ADE: case of the Netherlands. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences IV-4/W5, 2017, 73–81

Ledoux H, Arroyo Ohori K, Kumar K, Dukai B, Labetski A, Vitalis S, 2019. CityJSON: A compact and easy-to-use encoding of the CityGML data model. Open Geospatial Data, Software and Standards, 4:4

NEN, 2016. NEN 3610:2011/A1:2016 nl *Basismodel geo-informatie - Termen, definities, relaties en algemene regels voor de uitwisseling van informatie over aan de aarde gerelateerde ruimtelijke objecten* https://www.nen.nl/NEN-Shop/Norm/NEN-36102011A12016-nl.htm

Rouault E, 2014. GML madness. https://erouault.blogspot.com/2014/04/gml-madness.html, last visit: 2018–12–20

Eriksson, H., Harrie, L., and Paasch, J. M., 2018. What is the need for Building Parts? — A comparison of CityGML, INSPIRE Building and a Swedish building standard. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII- 4/W10:27–32.

ShapeChange (n.d.), Processing application schemas for geographic information, https://shapechange.net/

Sporny, M. G. Kellogg, and M. Lanthaler, 2014. JSON-LD 1.0: A JSON-based Serialization for Linked Data - W3C Recommendation, http://www.w3.org/TR/ json-ld/

Stoter, J. H. Ledoux, M. Reuvers, L. van den Brink, R. Klooster, P. Janssen, J. Beetz, F. Penninga and G. Vosselman, 2013. Establishing and implementing a national 3D standard in the Netherlands Photogrammetrie, Fernerkundung, Geoinformation 4, 2013, pp. 381–392.

## Appendix A Selection of 3D modelling requirements for IMGeo

| Requirement | Brief Description | Validation |
|---|---|---|
| 3.1 IMGeo 2.1.1 CityGML: Generic Requirements | | |
| Requirement 1 | The 3D data should be structured according to IMGeo-CityGML format. | Check this by using the developed validation tool |
| Requirement 2 | The IMGeo-CityGML data must comply with CityGML 2.0. In some cases we have more stringent requirements than CityGML | Check this by using the developed validation tool |
| Requirement 3 | Employ the EPSG 7415 Spatial Reference System (co ordinate system). | Check if the EPSG code 7415 is to be found in the CityGML file. |
| 3.2 Specifications for LOD0 Representation. | | |
| Requirement 4 | Every object in IMGeo is represented by a LOD0 geometry i.e. a TIN surface (triangulatedSurface) per object (tessellation of the object's footprint) . The LOD0 terrain is formed by a collection of such adjacent TIN surfaces, with recognizable object boundaries (constrained TIN) | Check if the number of polygons in LOD0 is the same as that in 2D IMGeo CityGML |
| Requirement 5 | The LOD0 geometries of all IMGeo polygons (water, road, building, land use, vegetation) at ground level should form a planar partition in 2.5D (no holes or overlap). | Check by looking for holes or overlap |
| Requirement 6 | The height difference between the terrain in reality and its representation in TINs is allowed to be maximum X cm. X can be dependent on the object type (for example another X can be chosen for hard surfaces with curbs than that for pasture). Individual apexes are acceptable until up to 3 times X, but connected pieces of a TIN of more than Y m2 may deviate no more than this X cm. | It can be requested that a colour coded point file be supplied in which the terrain points are coloured as a function of the height deviation with respect to the object surface that the terrain models. It can then easily be seen if areas (greater than Y m2) show greater deviation. |
| Requirement 7 | Vertical surfaces in the TIN may not occur, because many GIS software crashes on such data. Instead, vertical surfaces should be approached by maximum sloping surfaces. How this should be done depends on which objects are left and right of the vertical jump. The sloping surfaces need to be attached as follows to the relevant object: | Testing the Z component of the TIN triangles' normal vectors. These Z components may not be equal to 0. An alternative, but incomplete check, is to look for points with the same XY co ordinates, but the same Z co ordinates. |
| Requirement 8 | When very precise vertical intervals between specific objects are necessary, this should be recorded in the technical specifications. A minimum height should be defined and vertical intervals must be visible. Examples are the height jumps at the location of curbs. | Check randomly if small vertical intervals have been modelled. |
| Requirement 9 | Waterbodies are always flat, horizontal surfaces. | Testing the X and Y component of surfaces' normal vectors. These must be equal to 0. |
| Requirement 10 | IMGeo polygons which are above or below the terrain should be modelled with a triangulatedSurface which connects up to the topologically consistent ground level. The result is the stacking of 2.5 objects. | Overlapping objects with differing levels may not intersect each other in height. |
| Requirement 11 | All IMGeo polygons should be assigned to the IMGeo LOD0 representation, i.e. both those at ground level as well as the ones above and below ground level | Check if a number of polygons in LOD0 are in agreement with the polygons in 2D IMGeo CityGML. |
| Requirement 12 | Terrain Intersection Curves (TIC's) should be used in order to make ClosingSurfaces where 3D objects hang above or in the terrain model. This results in a closed topologically correct terrain model. | |
| 3.3 Building Specifications | | |
| Requirement 13 | The ground surface of a building at LOD1 and LOD2 must be horizontal. The ground surfaces should, though, be determined per individual | |

| | | |
|---|---|---|
| | building and not per block of buildings . This surface is then positioned at the lowest height of the terrain at the location of this surface so that the building sinks "in" the terrain and gaps between ground surface are avoided. | |
| Requirement 14 | Notwithstanding the CityGML specification, LOD0 footprint must be determined where the outside wall touches the terrain. | |
| Requirement 15 | An LOD1 representation should be supplied for every IMGeo building. | Easy to check if the building's IMGeo ID is saved as an attribute to the LOD1 representation. |
| Requirement 16 | The building height is the median of the height of the points which are positioned within the footprint. | Check randomly if the median of the height of the points on one roof lies within a margin of X cm from the height in the model. |
| Requirement 17 | If a building's roof has significant vertical intervals (for example a church with a tower), then these differing height levels should be distinguished in 3D, particularly if the interval is greater than, for example, 1.5 metres and if the surface area is greater than 4 square metres | |
| Requirement 18 | The lower surfaces of the building's block geometry must correspond to the 2D and LOD0 geometry in IMGeo. | |
| Requirement 19 | The lower surface of a LOD1 block should be horizontal, taking the lowest point of the footprint's terrain triangulation as its height (see LOD0 building) | |
| Requirement 20 | For buildings which bridge roads or water, through passage should be guaranteed. This may be artificially applied. | |
| Requirement 21 | The geometry of LOD1 Buildings should be defined in CityGML as GML:Solids (closed volumes, also from below) and not as GML:MultiSurface, which is permitted for LOD1 buildings.. | Each building object consists of exactly one solid. |
| Requirement 22 | Each LOD2 IMGeo building is modeled by the GML:Solid geometry type in which the semantics of the boundaries (surfaces) are made explicit (e.g. footprint, roof surface, wall surface). LOD2 buildings can be represented as a collection of a solid with other geometry types such as a multisurface for a roof overhang | Each building object consists of a minimum of one solid. |
| Requirement 23 | The locations of the outer walls of 3D building models should be in alignment with the 2D boundaries from the BGT and the BAG (preferably the BGT). | Randomly test if boundaries from the BGT or BAG have been taken up. |
| Requirement 24 | Roof boundaries of 3D building models are in agreement with 2D boundaries from the BGT or BAG (preferably the BAG). | |
| Requirement 25 | Building models should be complete in the sense that the combination of all of a building's surfaces collectively forms a closed volume, a 3D solid. No surface from another building may be positioned within a building model. Building models may touch each other, but not overlap. | Check by means of the developed validation tool. |
| Requirement 26 | When a roof overhang is explicitly modelled, roof surfaces should be split at the roof overhang's location in order to result in a solid geometry. These roof overhangs should be modeled as a (multi)surface and the rest of the roof should form a part of the solid geometry's boundary. | |
| Requirement 27 | When a roof overhang is explicitly modeled, roof surfaces should be split at the roof overhang's location to obtain a valid solid geometry. These roof overhangs should be modeled as a (multi)surface and the rest of the roof should form a part of the solid geometry's boundary | The supplier can be asked to supply a colour coded point cloud in which the points colour within BAG/BGT polygons a function is of the height difference with the modelled roof. Larger deviations can then be spotted easily. The surface area of each "connected component" of points which deviate too much can be calculated with a little more effort. |
| Requirement 28 | LOD2 roof surfaces with a minimum surface area of X m2 may not deviate more than Y m in height from the corresponding points from the point cloud | Checking can be done with a colour coded file, just as by the previous Requirement, although this time with a colour dependent on the angle difference between normal vectors which have been estimated |

| | | |
|---|---|---|
| | | from points which lie within a certain radius and normal vectors from the modelled surfaces. |
| Requirement 28 | Roof surfaces with a minimum surface area of X m2 may not deviate more than Y degrees in the normal direction from a surface because of the corresponding points from the point cloud. This prevents very flat saddle roofs to be modeled by flat roofs and mansard roofs to be modeled by saddle roofs | Check in the same way as Requirement 31. |
| Requirement 29 | Curved surface areas should be represented by a triangulation in which deviation between the true surface area and the triangulation is not more than Xm. | |
| Requirement 30 | Roof surface corner points in the model (for as much as they haven't been misaligned by the BAG) must lie within a distance of Xm from the closest neighbouring data points | Check if there are data points present within a radius of X m from a vertex (and within a BAG outline). Use a 3D query option or specialised software. |
| Requirement 31 | The solids of buildings in LOD1 and LOD2 should conform to the requirements which are discussed in 4.3.4. | Check by means of the developed validation tool. |