BIMy Project:
D2.2 Integrated BIM/GIS model

Document metadata

Date	2021-03-08
Status	Final
Version	2.00
Authors	Stijn Goedertier – GIM Bert Akkermans – GIM Bert Meeus - GIM Niels Gabriels - GIM Steven Smolders - GIM Elena Pajares - ASSAR Thomas Goossens – ASSAR Hashmat Wahid – Willemen Steven Smolders – GIM Niki Cauberg - BBRI
Coordinator	Stijn Goedertier/ Niels Gabriels – GIM
Reviewed by	Steven Smolders /GIM

Version history

Version	Date	Author	Change
0.01	2019-04-09	SGO	Introduction and scope
0.02	2019-04-10	SGO	IFC to 2D SVG floorplan
0.03	2019-05-22	SGO	Modeling time and scale in existing standards
0.04	2019-05-23	TGO	Feedback on the IFC to 2D SVG floorplan transformations
0.05	2019-05-28	SGO	IFC to CityGML 2.0
0.06	2019-06-07	SGO	GIS-to-BIM: feedback from the workshop with Belgian consortium members
0.07	2019-06-14	HWA	2.1 IFC and 2.2 Revit
0.08	2019-06-18	SMO	General review

0.09	2019-06-19	SGO	Georeferencing and measurement units in IFC
0.10	2019-06-20	SGO	IFC to 3D tiles
0.11	2019-06-26	SGO	Linking GIS features to BIM models
0.12	2019-06-27	SGO	Building Topology Ontology
0.13	2020-02-10	SGO/SS	Update of BIMy Data Model
1.00	2020-05-24	SGO	Version for Year 2 review
1.10	2020-09-25	SGO/BM/SS	Integration of LiDAR to IFC processing
1.20	2021-02-14	SS/NGA	Integration of KLIP/IMKL data – final confidential version
2.00	2021-03-10	SS/NGA	Public version

Table of Contents

1 Introduction

This document defines the required mapping rules to convert between the BIM and GIS data standards that are used in the BIMy platform. This chapter introduces the context, objective, scope, and methodology applied to write this document.

1.1 Context: BIMy as a collaborative platform

The BIMy project, as stated in the Full Project Proposal (BIMy consortium, 2017), aims at providing an **open collaborative platform for sharing, storing and filtering BIMs among different BIM owners/users and integrating and visualizing them in their built and natural environment**. BIMy can be seen as an open and generic intermediary that enables interactions between existing and new applications through a unique standardized open API platform. Such a platform will provide a secure collaborative working environment where different stakeholders can benefit and/or utilize BIM models not only at single building level but also at larger levels that can be scaled up to wider-area smart city applications.

BIMy will overcome the limitations of current BIM exchange platforms, providing the following features: **BIM with scale and time** (supporting different levels of details and different stages of the building lifecycle), **BIM/GIS semantic and dynamic integration** and **cloud storage** (integrating BIM in their built and natural environment), BIM filtering (providing relevant information according to stakeholders and applications), cooperation (supporting stakeholder interactions), simulation and 3D visualisation (mixed and augmented reality through different devices).

BIMy is bringing into the consortium all the actors necessary to the successful completion of the platform. There are **large companies** that can provide a **Cloud infrastructure** for hosting the BIMy platform and contribute with bigger resources when needed. The **smaller companies** offer more focused know-how to specified tasks as collaboration or BIM sharing and visualisation. The **research partners** will support companies with more complicated problems such as creating simple API and modelling and integrating BIM and GIS at different scales and times. **BIM owners/users** have an important role in definition of the requirements, modelling, in offering their expertise for different applications and business models as well as the evaluation of demonstrators. The demonstrators in two different countries improve the chances to make BIMy more replicable to new countries and environments. This enhances remarkably the market potential of BIMy.

1.2 Objective

As stated in the Final Project Proposal (BIMy consortium, 2017), the goal of this deliverable is to **formalize the semantic and geometric relationships between GIS and the extended Building Information Model (BIM)** (extended with time and scale, resulting from task T2.1). It will rely on the extended IFC obtained in T2.1. and CityGML. The goal is therefore not to come up with a new standard integrating IFC and CityGML, but rather to formally identify the relationships between concepts in the two standards.

Additionally, this task will specify the data requirement of BIMy in regard of existing and ongoing standardisation, open data and linked data initiatives.

1.3 Scope

This document formalizes the semantic and geometric relationships between data standards and formats. Its scope is limited as follows:

- Only data standards that are *used* in the BIMy platform will be considered.
- Mapping relationships will be defined in a *unidirectional* manner, only when a data transformation is needed.
- Mapping relationships will be formalized at the level of classes and geometry types. Detailed mappings at the level of attributes or values will not be provided.
- Mapping relationships will be formalized but will not necessarily be *executable*. The implementation of the transformation as an executable transformation is part of work package 3.

Table 1 Semantic and geometric mapping matrix

FROM \ TO	IFC	Floorplan (SVG)	Revit	Cesium3DTiles	CityGML	BOT	GeoTiff	GML	LiDAR	Shape	FilmBox	Unreal Datasmith	OBJ/Material (AR)
IFC		√		√	√	√					√	√	√
Floorplan (SVG)													
Revit	√			√							√	√	√
Cesium3DTiles													
CityGML	√												
BOT													
GeoTiff	√		√										
GML (IMKL)	√		√	√	√								
LiDAR (LAS)	√		√	√	√		√				√	√	√
Shape	√		√	√	√						√	√	√
DEM	√			√							√	√	√
IMKL	√			√							√	√	√

1.4 Methodology

To formalize the semantic and geometric relationships between data standards we will apply the following methodology:

- We first study the data standards and summarize their main characteristics. In particular, we will look at time and scale aspects, but also at georeferencing for BIM standards. This is documented in Section 2.
- We then study the transformation between a selected set of data standards. For each transformation, we will list the requirements, analyse existing implementations, summarize the transformation rules, develop an executable implementation, and list future improvements. This

is documented in Section 3 for BIM-to-GIS data transformations and Section 4 for GIS-to-BIM data transformations.

2 Integrating BIM and GIS data standards

This section briefly introduces the data standards used in the BIMy platform. For each standard, it briefly analyses the capabilities of representing time and scale. The intent is to make the model transformations described in the next section more comprehensive.

Looking at the difference between GIS and BIM typically the following aspects, summarized in Figure 1:

- **Scale**: whereas BIM represents objects at high-scale, GIS commonly looks at a much lower level of detail.
- **Coordinates**: in BIM, it is more common to use a local reference point and Cartesian coordinates (x,y,z) using various measurement units (meter, centimeter, millimeter). In GIS, geodetic coordinates are used in relation to a reference ellipsoid and a geodetic datum, defined by *standardized* geospatial coordinate reference systems. To represent a location on a plane, a *map projection* is used, enabling the use of Cartesian coordinates and metric measurement units.
- **Data source**: IN BIM, data is created by human design effort (BIM modeler), whereas in GIS, surveying techniques are more commonly used for data gathering.

Figure 1 Differences between GIS and BIM

2.1 IFC

Industry Foundation Classes (IFC) is a standardized file format (ISO 16739 standard) used by the construction industry to exchange and share information between software applications. Developed by BuildingSmart, the IFC is intended to be the interoperability format for OpenBIM.

IFC is rarely used as a *native* data format in BIM tools, but rather as an exchange format. BuildingSmart France (bimstandards.fr, n.d.) keeps track of a number of conversion procedures on the use of IFC in several commercial BIM tools. Also, Willemen has on its website a number of guidelines, for example for the export of IFC from REVIT.

Like CityGML, IFC allows for multiple geometric representations, via the IfcProductDefinitionShape and IfcLocalPlacement. A detailed specification for the shape representation is introduced at the level of subtypes of IfcElement. This is not possible in Revit. There is no “LOD” attribute linked to every object. It is possible to add this information in a parameter of the object (manually). But the same object cannot be

visualized depending on different LOD parameters. In Revit, it is possible to update the family on the BIM Model (e.g., change a part of the model from LOD400 to LOD300), but both versions together cannot exist in the same model.

2.1.1 Georeferencing in IFC

The IFC standard provides several IFC entities that allow to describe the required information for its geo-referencing:

- **IfcSite**: uses the RefLatitude, RefLongitude, and RefElevation attributes. The latitude and longitude are defined as angles with degrees, minutes, seconds and optionally millionths of seconds with respect to the WGS84 (EPSG:4326) coordinate reference system. It is also possible to use IfcCoordinateReferenceSystem and IfcMapConversion to use a specific coordinate reference system.
- **IfcGeometricRepresentationContext (IFC 2x3)**: the WorldCoordinateSystem attribute can be used to offset the local coordinate system of the project's point of origin from the global geodetic datum. The TrueNorth attribute can be used to rotate the BIM model, if the y axis of the *WorldCoordinateSystem* does not point to the global northing. If an geographic placement is provided using IfcMapConversion then the *WorldCoordinateSystem* attribute is used to define the offset between the zero point of the local engineering coordinate system and the geographic reference point to which the IfcMapConversion offset relates. In preferred practice both points (also called "project base point" and "survey point") should be coincidental. However, it is possible to offset the geographic reference point from the local zero point.

Note that IFC does not oblige using a local coordinate system with local reference point. It is possible (even desirable) to us a 2D geospatial reference (e.g., Belgian Lambert 72 EPSG:31370) and a common vertical reference system (e.g. Belgian TAW).

In practice, tools are poorly supporting these entities, or these entities are not filled in at all. When multiple partners collaborate in a construction project, they establish a BIM protocol that defines the coordinates system and reference point used throughout the project. Although it is possible that partners use wrong georeferencing or make exports without information about georeferencing in the IFC. Because of this poor practice, Clemen and Görne define a **Level of Georeferencing** (LoGeoRef) to measure the quality of georeferencing (Clemen & Görne, 2019) in IFC files. The have developed an open-source tool, IfcGeoRefChecker, to extract the georeferenced information and asses the quality level. Figure 2 shows a (simplified) output of this tool.

Figure 2 Output of the IfcGeoRefChecker tool for the ASSAR Kortrijk model

```
Examination of 015.10_CM_Architectuur_Stabiliteit_optimized.ifc regarding georeferencing
content.

IfcVersion: Ifc2X3
LengthUnit: mm

Geographic coordinates referenced in #107334=IfcSite:
 Latitude: 50.8157424925
 Longitude: 3.26075148555556
 Elevation: 0
```

```
Local placement relative to the WorldCoordinateSystem
 Referencing Element:#107334=IfcSite
 Placement referenced in #912279=IfcAxis2Placement3D
  X = -35880.4631372896
  Y = -33693.3508431482
  Z = 0
 Rotation X-axis = (-0.687214311541541/0.726454740512088/0)
 Rotation Z-axis = (0/0/1)

Project context attributes for georeferencing (Location: WorldCoordinateSystem / Rotation:
TrueNorth)
 Project:#1109794=IfcProject
 Project context element: #1115=IfcGeometricRepresentationContext
  X = 0
  Y = 0
  Z = 0
  Rotation X-axis = (1/0/0)
  Rotation Z-axis = (0/0/1)

 True North:
  X-component =6.12303176911189E-17
  Y-component =1
```

Within the IFC is possible to alter this information by editing the IFC in notepad.

When we adjust the IFCs manually, we must take the units in which we have to work into account. These units can be found at the beginning of the IFC under IFCSIUNIT:

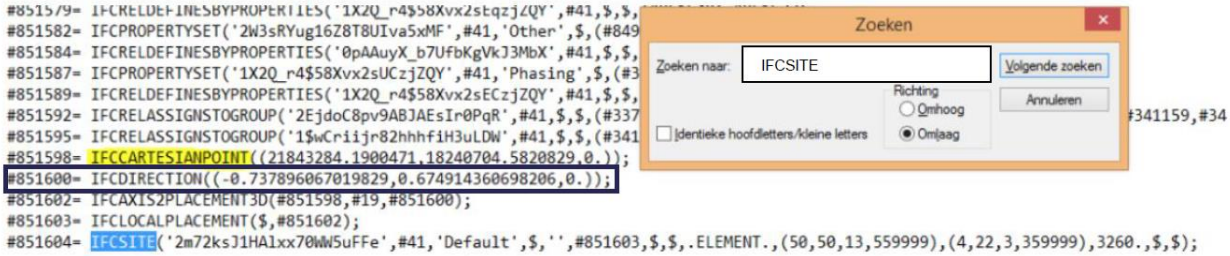
```
#42= IFCSIUNIT(*,.LENGTHUNIT.,.CENTI.,.METRE.);
#43= IFCSIUNIT(*,.LENGTHUNIT.,$,.METRE.);
#44= IFCSIUNIT(*,.AREAUNIT.,$,.SQUARE_METRE.);
#45= IFCSIUNIT(*,.VOLUMEUNIT.,$,.CUBIC_METRE.);
#46= IFCSIUNIT(*,.PLANEANGLEUNIT.,$,.RADIAN.);
#47= IFCDIMENSIONALEXPONENTS(0,0,0,0,0,0,0);
#48= IFCMEASUREWITHUNIT(IFCRATIOMEASURE(0.017453
#49= IFCCONVERSIONBASEDUNIT(#47,.PLANEANGLEUNIT.
#50= IFCSIUNIT(*,.MASSUNIT.,.KILO.,.GRAM.);
```

<u>IFC exported from REVIT</u>

Subsequently we search for IFCSITE. THE IFCCARTESIANPOINT that is closest to IFCSITE contains the reference point of the model. We don't search for IFCCARTESIANPOINT as there are many IFCCARTESIANPOINT in the IFC.

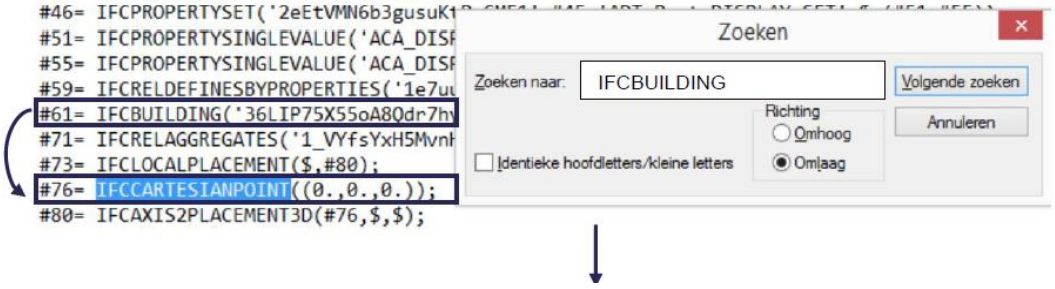
```
#851582= IFCPROPERTYSET('2W3sRYug16Z8T8UIva5xMF',#41,'Other',$,(#849
#851584= IFCRELDEFINESBYPROPERTIES('0pAAuyX_b7UfbKgVkJ3MbX',#41,$,$,
#851587= IFCPROPERTYSET('1X2Q_r4$58Xvx2sUCzjZQY',#41,'Phasing',$,(#3
#851589= IFCRELDEFINESBYPROPERTIES('1X2Q_r4$58Xvx2sECzjZQY',#41,$,$,
#851592= IFCRELASSIGNSTOGROUP('2EjdoC8pv9ABJAEsIr0PqR',#41,$,$,(#337
#851595= IFCRELASSIGNSTOGROUP('1$wCriiic82hhhfiH3ulDW',#41,$,$,(#341
#851598= IFCCARTESIANPOINT((21843284.1900471,18240704.5820829,0.));
#851600= IFCDIRECTION((-0.737896067019829,0.674914360698206,0.));
#851602= IFCAXIS2PLACEMENT3D(#851598,#19,#851600);
#851603= IFCLOCALPLACEMENT($,#851602);
#851604= IFCSITE('2m72ksJ1HA1xx70WW5uFFe',#41,'Default',$,'',#851603,$,$,.ELEMENT.,(50,50,13,559999),(4,22,3,359999),3260.,$,$);
```

It is also possible to rotate the model by searching for IFCDIRECTION and enter the angular rotation



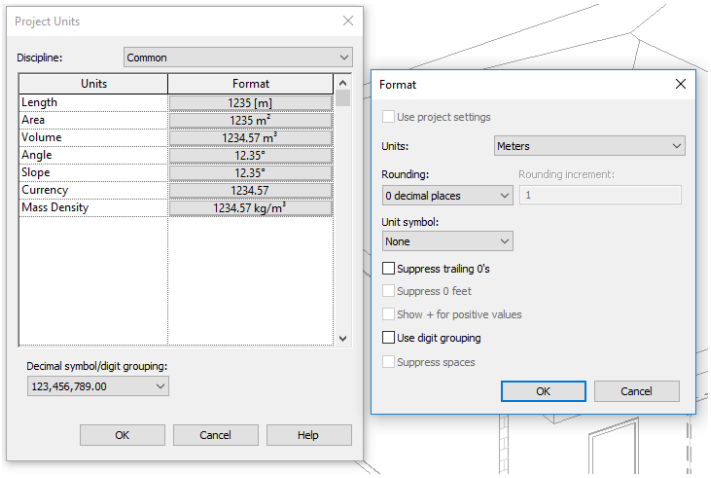
<u>IFC exported from another design software</u>

To find the right IFCCARTESIANPOINT, we search for IFCBUILDING. The reference coordinates can then be completed.



2.1.2 Measurement units in IFC

For example, when exporting from Revit to IFC, Revit does not allow to choose in which unit it should be exported. The IFC automatically gets exported to the units defined in the files project units. It is possible to change the projects unit when needed. In most BIM software it doesn't matter in which units the IFC is exported. For example, in solibri with 2 IFCs with different IFCSIUNIT.



2.2 Revit

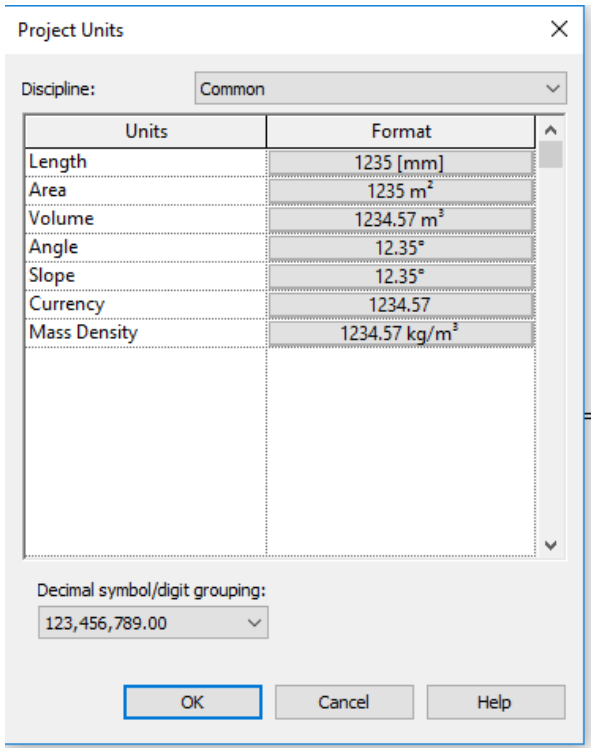
2.2.1 Georeferencing Revit

Revit uses three reference points: an "Origin Point /Project Startup Location, a "Survey Point" and a "Project Base Point". The reason for the existence of different reference points is that two axis systems can be used to link with other models and files, a world axis system and a local coordinate system.

- **Origin Point / 'Project Startup Location'**: The startup location, the (invisible) zero point of a Revit project.
- **Survey Point**: This point represents a known point in the real world, for example a measuring point. In the case of a local coordinate system this point is in fact unnecessary.
- **Project Base Point**: This is a known point of the model. It is a "unique position" within each model, a kind of "internal origin". It is the reference point of the local coordinate system.

2.2.2 Measurement units in Revit

A new project is started with the following units. This can be easily changed.



2.3 3D Tiles

OGC 3D Tiles (OGC, 2019b) is a specification that is originally developed by Analytical Graphics Inc. (AGI) and that allows for efficiently streaming 3D geospatial data on the Web. The format is often used in combination with the CesiumJS viewer. The specification has accepted as an OGC Community Standard.

The specification used a 3d geometry model based on glTF. Each tile described by one JSON file with CRS, attribute, and refinement information. An alternative to 3D Tiles is the Indexed 3D Scene Layer Format (I3S), another OGC Community standard submitted by ESRI amongst other organisations. To note is that OGC is meanwhile working on the "GeoVolumes" API as the 3D extension of the OGC API for serving tiles that is better aligned with the ISO/OGC information models and the 2D content delivery standards.

The ongoing work on Time-Dynamic 3D Tiles will allow representing time-varying features. In a blog post Pascal Poulain (Poulain, 2018) reports on experiments including 3D weather data time series from GRIB files represented as 3D tiles with time interval properties. A CesiumJS viewer application is callable of visualizing changing weather data as the clock advances. He reports performance issues with regards to the current inability of the viewer to retrieve *tiles* based on a given time-range filter.

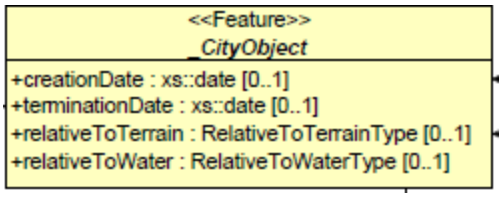
The 3D Tiles specification provides several mechanisms to cope with a changing level of detail, in accordance with the zoom-level of the application.

- **Hierarchical Level of Detail (HLOD) and refinement**: Tiles are structured into a tree incorporating Hierarchical Level of Detail (HLOD) so that at runtime a client implementation will need to determine if a tile is sufficiently detailed for rendering and if the content of tiles should be successively refined by children tiles of higher resolution. An implementation will consider a maximum allowed Screen-Space Error (SSE), the error measured in pixels. Refinement determines the process by which a lower resolution parent tile renders when its higher resolution children are selected to be rendered. Permitted refinement types are replacement ("REPLACE") and additive ("ADD"). If the tile has replacement refinement, the children tiles are rendered in place of the parent, that is, the parent tile is no longer rendered. If the tile has additive refinement, the children are rendered in addition to the parent tile.
- **Instanced 3D Model**: Instanced 3D models is a tile format for efficient streaming and rendering of a large number of models, called instances, with slight variations. In the simplest case, the same tree model, for example, may be located—or instanced—in several places. Each instance references the same model and has per-instance properties, such as position. This greatly reduces the data size of 3D tiles and will positively affect rendering times.

2.4 CityGML 2.0

CityGML 2.0 (OGC, 2012) is an XML data standard by the Open Geospatial Consortium (OGC). CityGML is modelled as an Application Schema of the Geography Markup Language (GML).

To represent time, the CityGML 2.0 encoding standard provides a **creationDate** and a **terminationDate** property. Yet, these properties may fall short. To cater for additional information modelling needs, the CityGML offers an **Application Domain Extension (ADE)** mechanism: depending on the specific needs, new features or properties can be added, greatly enhancing the model capabilities of CityGML. For example, the Time-Dependent Variables CityGML Application Domain Extensions (ADE) already include temporal aspects (i-SCOPE Project, 2013) to model time-varying attributes such as room temperature. The latter is similar to the IfcTimeSeries.



To represent scale, GIS standards like CityGML 2.0 allow representing an object with ***more than one geometry***. This is depicted in Figure 3 where the class AbstractBuilding is linked to more than one geometric representation.

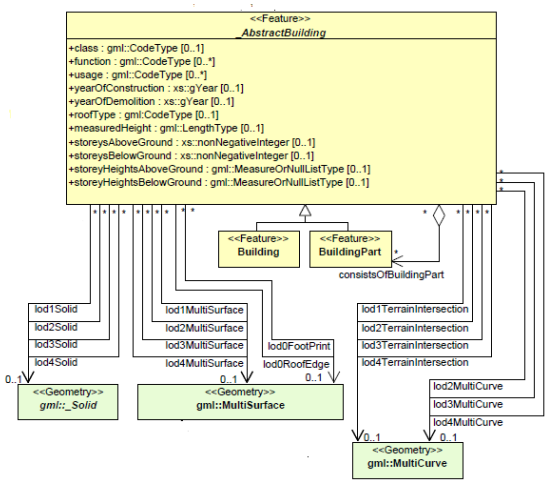


Figure 3 A CityGML 2.0 AbstractBuilding can have more than one geometric representation.

2.5 CityGML 3.0

The CityGML 3.0 conceptual model (available for public comments since December 2020) (OGC, 2019a) is looking into properties such as "validFrom" and "validTo" that refer to the lifespan a specific *version* of an object exists in the real world (this is taken from the INSPIRE Generic Conceptual Model) and "dateOfConstruction" and "dateOfDemolition" refer to the lifespan the whole object exists in the real world.

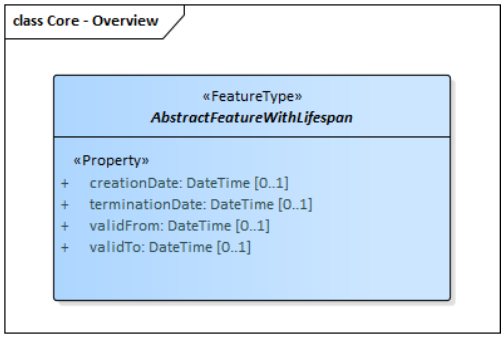


Figure 4 CityGML 3.0 Core (under development) time attributes

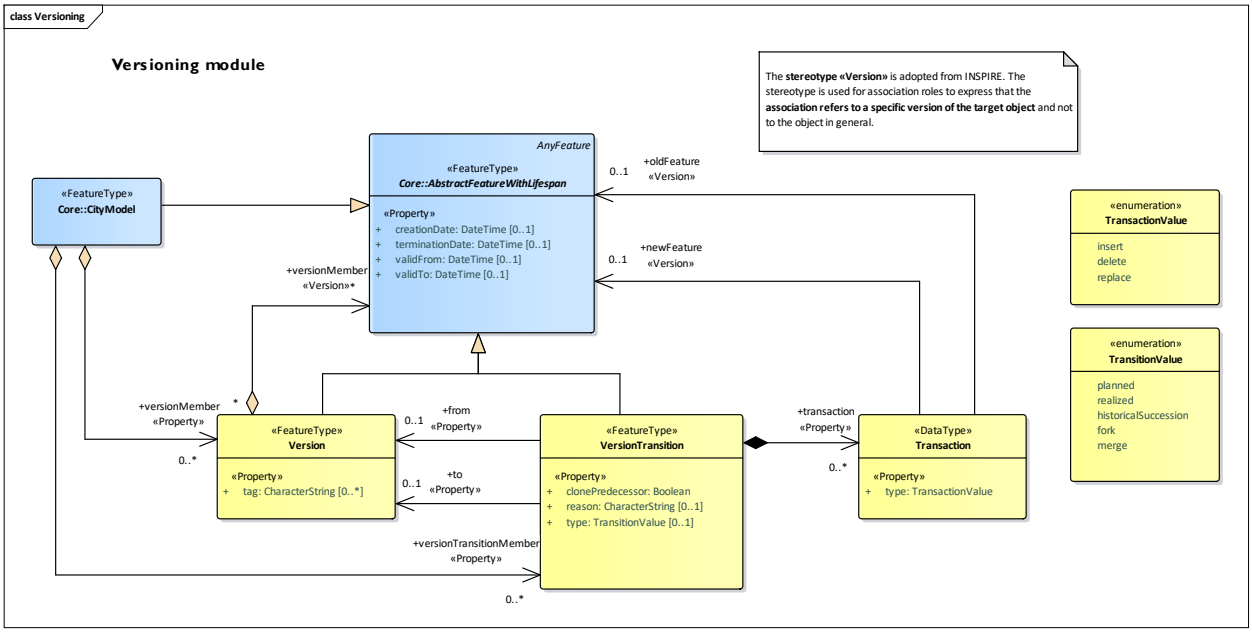


Figure 5 CityGML 3.0 Versioning module

Similar to the Time-Dependent Variables CityGML Application Domain Extensions (ADE) already include temporal aspects (i-SCOPE Project, 2013) for CityGIM2.0, CityGML3.0 – under development at the time of writing this deliverable – has included a similar model into the main specification.

€

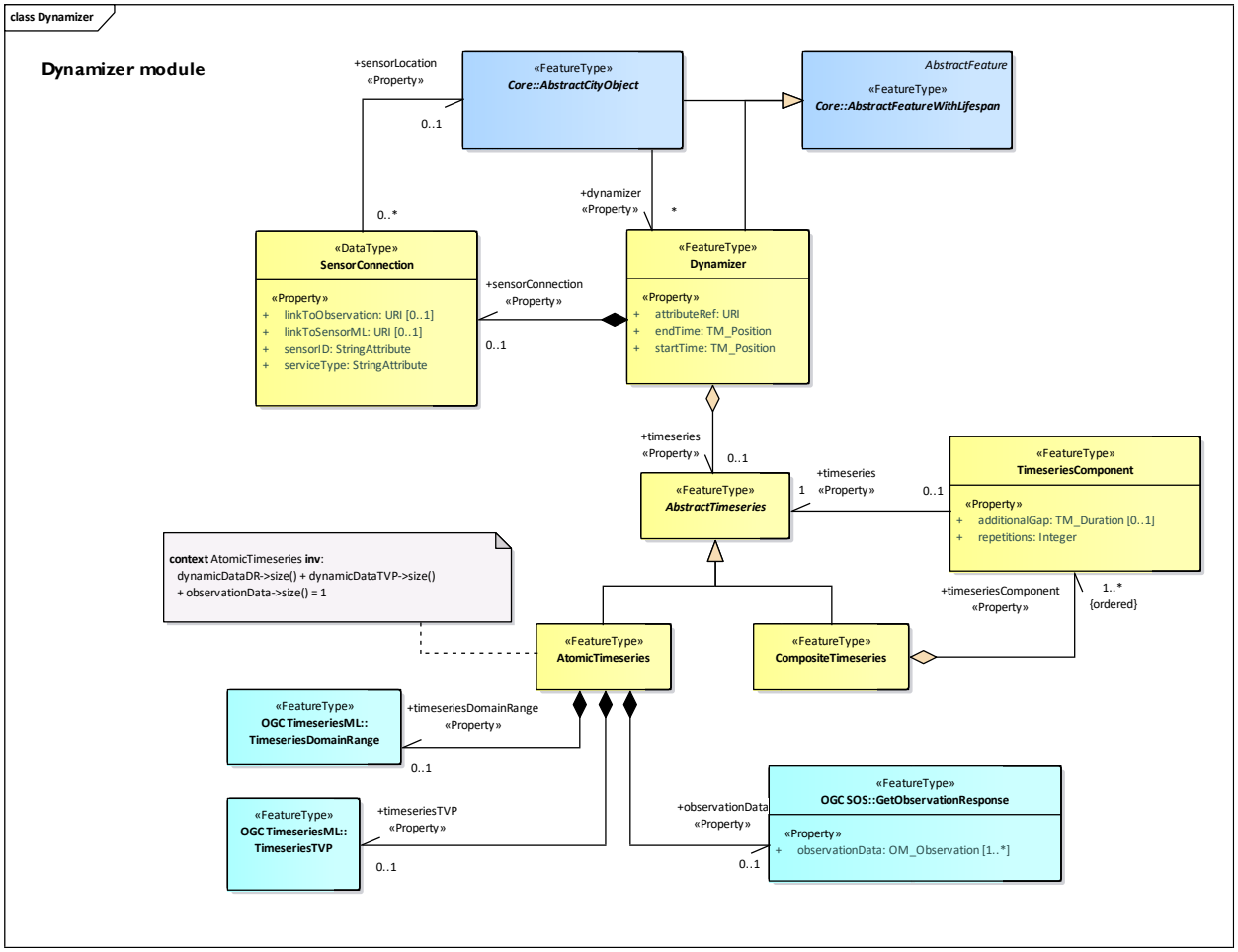
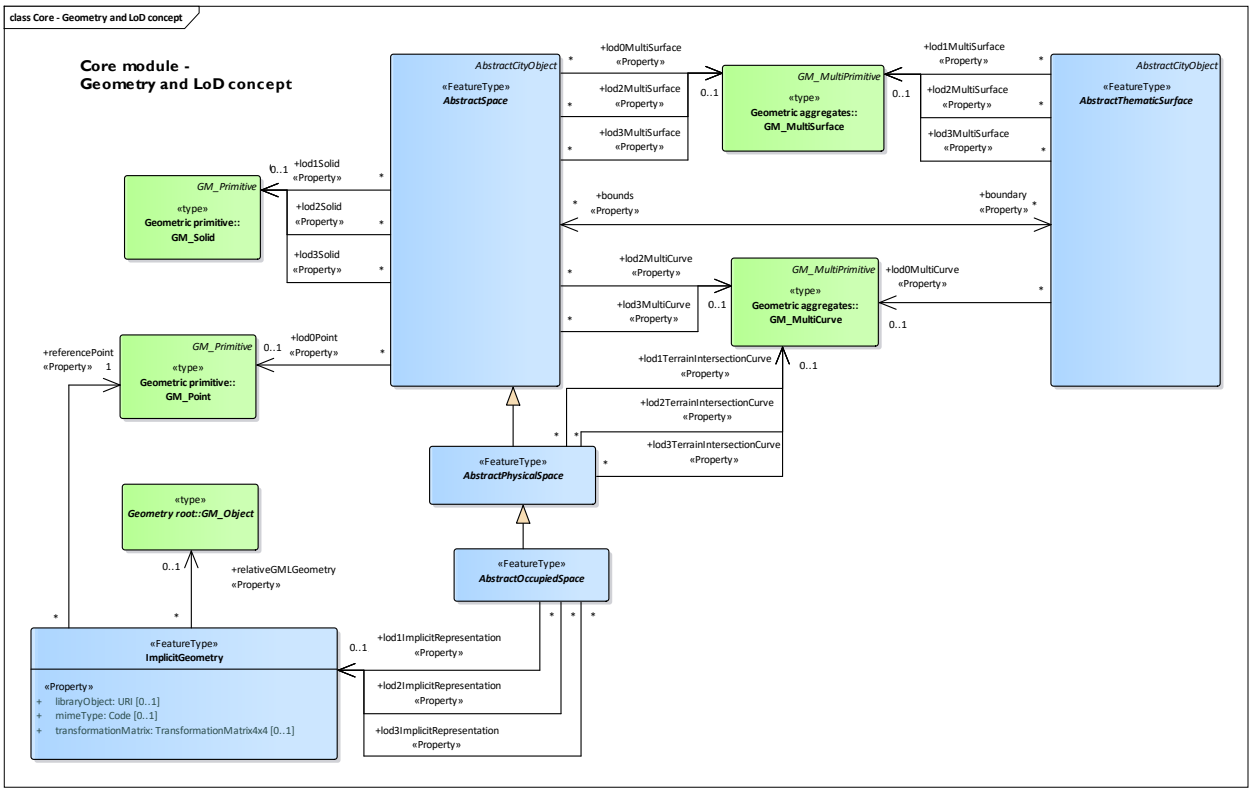


Figure 6 CityGML 3.0 Dynamizer module (under development)



2.6 Building Topology Ontology (BOT) and Ontology for Property Management (OPM)

The World Wide Web Consortium (W3C) Linked Building Data (LBD) Community Group has produced a number of data models – called ontologies – inspired by the IFC data model, but with a simpler structure. Such ontologies include the Building Topology Ontology (BOT) (W3C, 2019) and the PROPS ontology. The University of Ghent has developed an IFC-to-building-data convertor (Bonduel et al., 2018).

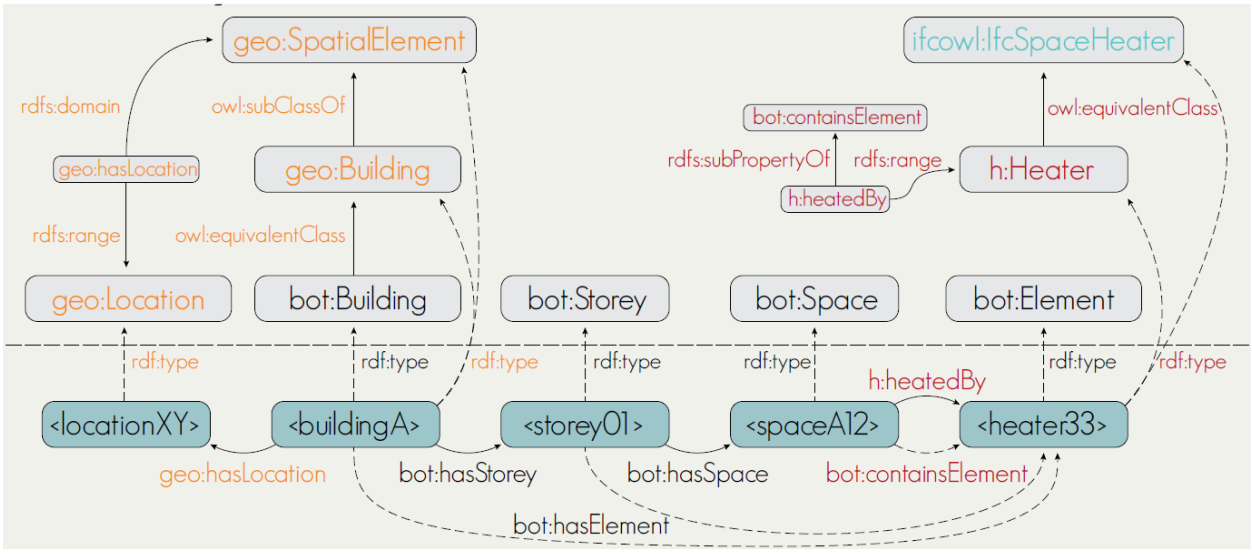


Figure 7 Illustration of the Building Topology Ontology (BOT) (Terkaj et al., 2017)

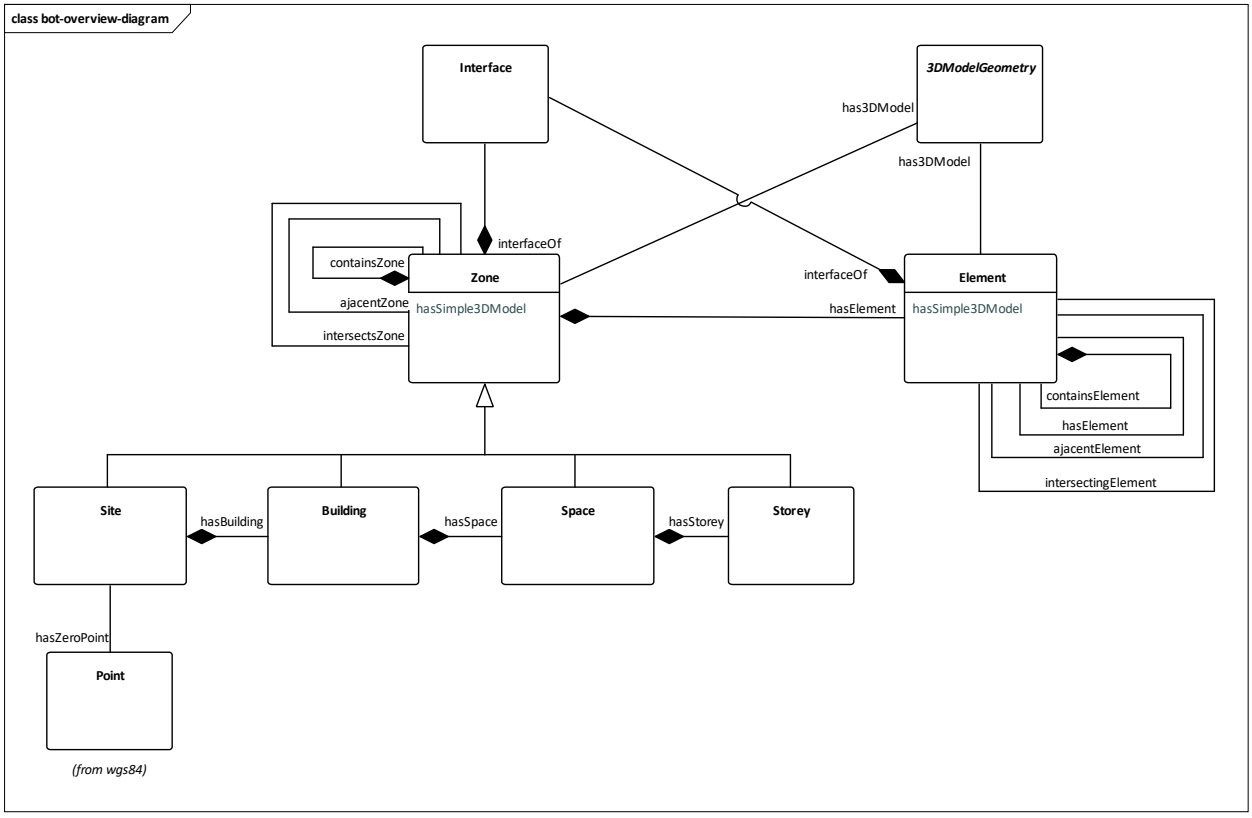


Figure 8 UML Class Diagram representation BOT

The Ontology for Property Management (OPM) is an ontology for describing temporal properties that are subject to changes as the building design evolves.

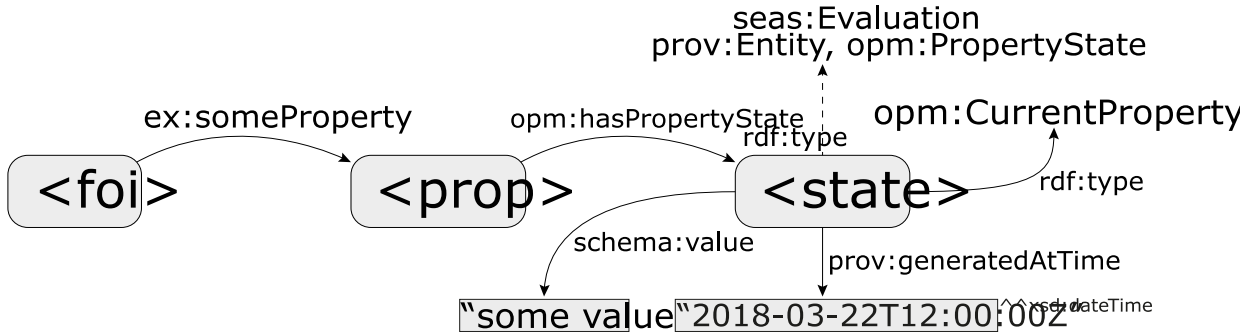


Figure 9 Time-varying properties can be expressed in the Ontology for Property Management (OPM)

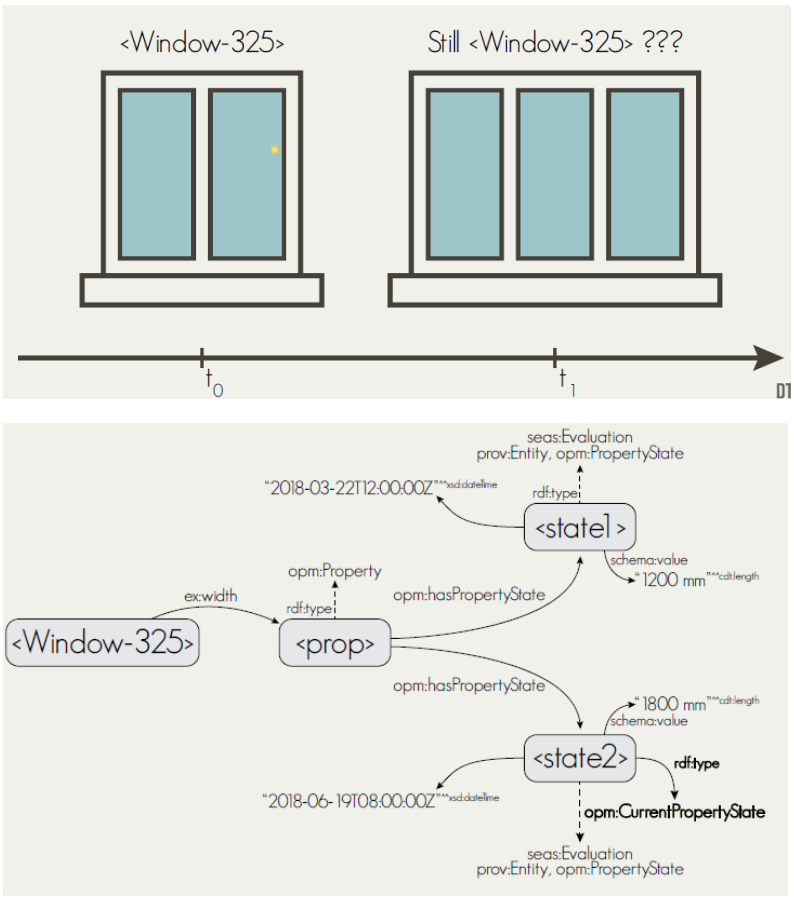


Figure 10 An illustration of time-varying properties in OPM (Holten Rasmussen, 2018)

3 BIM-to-GIS Mappings

This section formalizes the semantic and geometric transformations that are needed between the BIM and GIS data standards that are used in the BIMy platform. The starting point will be the BIM data in IFC or REVIT data formats. The next chapter will look at GIS data formats as a starting point.

3.1 IFC to 2D SVG floorplan

This section describes the model transformation that is needed to convert an IFC BIM model into a 2D floorplan.

3.1.1 Related requirements

This translation relates to the following requirements described in deliverable D1.2 (BIMy consortium, 2019a):

- Req_Y1_041: Visualisation of floor plans by Fire Brigade
- Req_Y1_039: Transform 3D digital model into a 2D floor plan
- Req_Y1_057: Visualisation of a simplified 3D IFC Model for the fire brigade

3.1.2 Transformation rules

To generate a 2D floorplan, the following transformations are required:

- **Remove irrelevant IFC objects**: less relevant objects of type IfcSpace, IfcCeiling, etc. must be filtered out.
- **Split objects by floor**: in IFC, most objects are assigned to a floor level (IfcBuildingStorey)
- **Reduce to a 2D representation**: the 3D geometry of each individual object must be reduced to a 2D form. This can be done in the following ways:
    - **Take a cross-section** at a certain elevation (e.g., 1m above floor level).
    - **Reduce the z-axis** of all coordinates and clean up redundant lines.
- **Apply formatting rules:** The resulting polygons and lines need to be colored according to the required semantics. Where needed, labels can be added, for example the name property of an IfcSpace.
- **Filter and/or rename object attributes:** The IFC attributes can be reduced and/or renamed as required.
- **Write to SVG**: write the result to SVG. Ideally, the coordinates as read from the IFC file are preserved.

3.1.3 Implementation

As part of deliverable D3.2, GIM has implemented a transformation. Figure 12 and Figure 13 depict a screenshot of a Web application to visualize the floorplan in SVG format the Schependomlaan (openBIMstandards, 2014) and Kortrijk datasets (ASSAR, 2019).

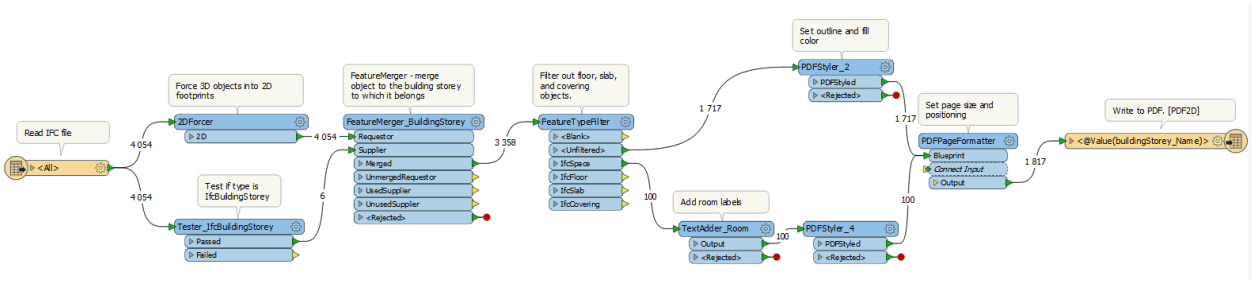


Figure 11 An implementation of the IFC-to-2D-SVG floorplan transformation

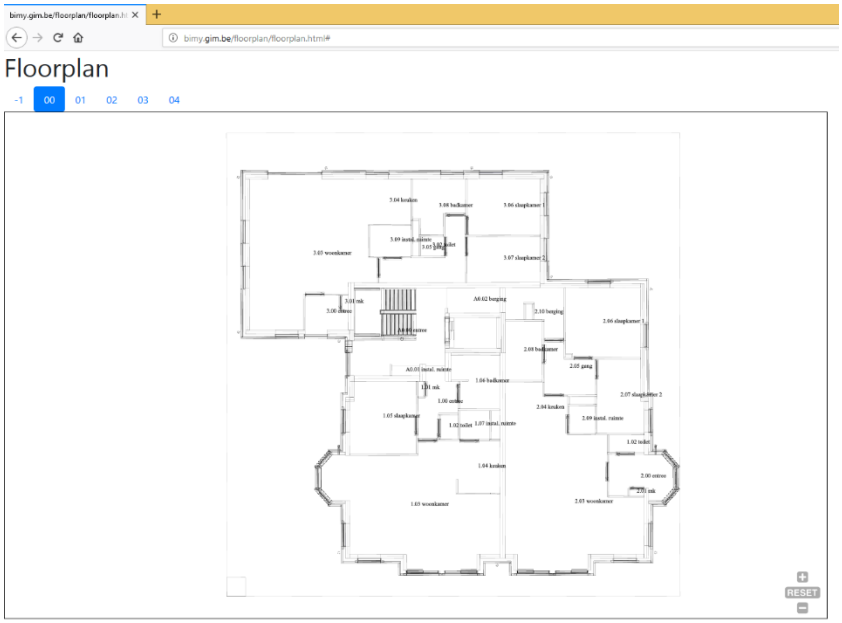


Figure 12 Screenshot of a Web application to visualize the floorplan in SVG format (dataset: Schependomlaan)

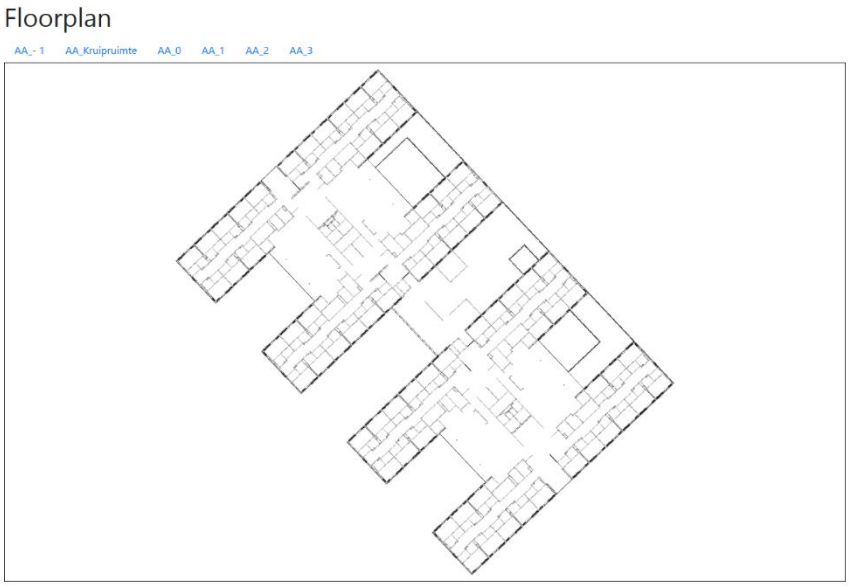


Figure 13 Screenshot of a Web application to vlisualize the floorplan in SVG format (dataset)

3.1.4 Future improvements

In the context of the Fire Prevention demonstrator, a number of improvements were identified.

- **Complex object hierarchy**: IFC uses a complex object hierarchy, when filtering the objects by floor level, the current implementation assumes that each object is a direct parent of an IfcBuildingStorey instance, which is not always the case.
- **Extract information directly from Revit:** Currently Assar Architects creates the Fire Prevention Plans directly from REVIT. These plans contain all the information that the fire brigade wants if they need a quick and clear overview of all the fire prevention aspects of a building. The information included in the plans is stored in two ways in Revit:
    - **3D-elements with parameters:**
        - Fire resistance of general building elements is stored in two parameters:
            - "FI_R": structural fire resistance of a load-bearing element (column, load-bearing wall, floor slab, etc.)
            - "FI_EI": temperature and flame resistance of structural elements (column, wall, slab, etc.) and non-structural elements (non-structural walls, etc.)

          The value of these parameters is expressed in minutes: usually 30, 60 (or 120)

          Fire prevention plans usually only display the EI-value of elements in drawing by displaying the wall in a certain color (e.g., EI30 = red, EI60 = green, EI120= blue). To avoid confusion, the R-value is usually not displayed visually on plan, only explained in text. (e.g., "Alle structural elements above ground level have R60, under ground level R120.")

        - Staircases: all staircases are modeled in 3D.

  - Indoor hydrants: are modeled in 3D.
  - Doors: Are modeled in 3D and contain all necessary information (EI-value + closing mechanisms) as parameters.
- **2D annotations added in plan views**: Both IFC2X3 and IFC4 may include 2D annotations. With the right parameters, Revit is able to export 2D geometry that remains linked to the desired view. It is also possible to export the classification of the 2D element exported. This can be used to recognize fire escape routes. The disadvantage is that the programs do not currently import these data correctly (Tested import with Revit, SimpleBIM, Solibri and Archicad). Compartments and subcompartments (areas), with surface area tag. Evacuation route (drawn as an area or filled region). Icons for location of hydrants, fire extinguishers, exit doors, evacuation exit doors and smoke ventilation openings.

To make these improvements, it would be best to start with the basics for the creation of the fire prevention plans.

1. First, make sure the plan is correct, complete and readable as a pdf generated from the IFC.
2. Then, add the general fire info that is contained in the model: Fire resistance of general elements (FI_EI), highlighted staircases, doors, hydrants.
3. After that, we can try to move on to the trickier ones: how to get the 2D info in the IFC model.

3.1.5 IFC Filtering: IFC to 3D IFC floorplan

As part of the Fire Risk Application, another approach was used: namely to create subsets of the BIM IFC model by floor. This “filtered” IFC model (filtered by building story, excluding e.g., ceilings and slabs) can then be visualized in viewers such as BIMsurfer3

Figure 14 IFC to 3D IFC floorplan visualized with BIMsurfer3 (ASSAR, 2019)

3.2 IFC to CityGML

This section describes the model transformation that is needed to convert an IFC BIM model into a CityGML counterpart.

3.2.1 Related requirements

This translation relates to the following requirements described in deliverable D1.2 (BIMy consortium, 2019a):

- UCS_Y1_08: BE: Integrate urban context information into BIM models
- Req_Y1_069: Transform BIM IFC model into CityGML

3.2.2 Existing implementations

The conversion of IFC to CityGML has been studied in the research community over the course of a decade. Nagel and colleagues (Nagel C et al., 2009) provide a detailed analysis of the conceptual similarities and mismatches in the conversion between IFC and CityGML 2.0. Figure 15 illustrates some of the most important differences:

- **Geometry representation**: Geometries in IFC are typically represented using Boundary Representation (BRep) as implemented in the STEP modelling language (ISO, 2018) and Constructive Solid Geometry (CSG). In CityGML, geometries are often reduced to surfaces. This is illustrated in Figure 15 (Nagel C et al., 2009). Also Ohori and colleagues report that the conversion between IFC and CityGML geometries is far from straightforward (Arroyo Ohori et al., 2017).
- **Coordinate reference system**: Geometries in IFC typically have engineering coordinates (x,y,z according to a locally defined zero point) whereas CityGML uses a wide variety of **geodetic coordinate reference systems**, that use a standardized zero point (a geodetic datum), and

possibly a projection. It is possible to include a georeferenced in an IFC file using the IfcSite element –A single reference point with WGS84 coordinates (Longitude, Latitude and Elevation) can be defined–This point defines the origin (0,0,0) of the LocalPlacement of all other objects which are hierarchically below, e.g. IfcBuilding relative positioning w.r.t. the local system of the site element

- **Hierarchical relationships**: Objects have complex hierarchical relationships. For example, in IFC Doors are the children of Openings, which in turn are the children of Walls, while in CityGML, Doors are children of Walls.

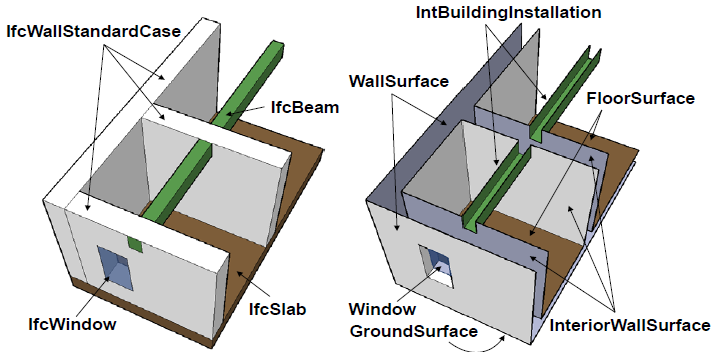


Figure 15 Comparison between IFC and CityGML (Nagel C et al., 2009)

In The Netherlands, the GeoBIM Benchmark project (Noardo et al., 2019) **provides a framework** describing the present ability of existing software tools to **use CityGML and IFC models** and understand their **performance** while doing so, both **in terms of information loss** and in terms of **ability to handle large datasets**. The benchmark defines specific tasks to test the following:

- Task 1 - Support for IFC within BIM (and other) software;
- Task 2 - Options for geo-referencing BIM data;
- Task 3 - Support for CityGML within GIS (and other) tools;
- Task 4 - Options for conversion (software and procedural) (both IFC to CityGML and CityGML to IFC).

3.2.3 Transformation rules

A transformation from IFC to CityGML 2.0 encompasses among others the following transformations:

- **Reading IFC objects**: all IFC objects must be read. The scale can best be converted.

```
#24=IFCSIUNIT(*,.LENGTHUNIT.,$,.METRE.);
#28IFCUNITASSIGNMENT((#24));
```

- **Transformation of geometries**: resolving the solid geometries (CSG or BRep) and convert geometries into MultiSurface representations.
- **Georeference geometries**: convert the Cartesian coordinates of the IFC file into geospatial coordinates according to a reference geometry. CityGML uses a 3D geodetic coordinate reference system. Because IFC files may use local coordinate systems, this may require the following adjustments:

  - **Scale:** adjust the unit of measurement, e.g. convert from millimeter into meter.
  - **Rotate:** it may be needed to rotate the objects. This is particularly needed if the y-axis is not directed at the true north.
  - **Offset**: It may be needed to offset the x,y coordinates in the IFC file according to the required translation defined in the IFC file's IfcGeometricRepresentationContext WorldCoordinateSystem attribute. It may also be needed to adjust elevation, for instance, if a different vertical reference system is used (e.g. Belgian TAW versus Dutch NAP).
  - **Reproject**: after applying the above-mentioned scaling, rotation, and offset, the coordinates of the IFC file may be in a geodetic coordinate reference system.
- **Readjust object hierarchies**: in IFC, many objects have an IfcBuildingStorey as direct parent. Other objects have a more complex hierarchy.
- **Construct aggregates**: In IFC, some instances are composed of other instances. For example, curtainwalls contain plates and members; stairs contain railings, stair flights and slabs; and ramps contain slabs and ramps. These parts must be added to their parent feature when converting to CityGML.
- **Rename object properties**: for instance, the IFC unique id of each object can be used as gml:id attribute in CityGML.
- **Change instance classification**: convert the IFC classification of instances into a corresponding CityGML feature type, Table 2 gives an indication of this.

Table 2 Mapping IFC entities to CityGML feature types

IFC entities	**CityGML 2.0 feature types**
IfcProject	CityModel
IfcBuilding	Building
IfcSpace	Room
IfcRoof	RoofSurface
IfcFloor	FloorSurface
IfcWallStandardCase	WallSurface
IfcWall	WallSurface
IfcCurtainWall IfcMember	WallSurface
IfcSlab	FloorSufrace
IfcDoor	Door
IfcWindow IfcPlate	Window
IfcRailing	BuildingInstallation
IfcColumn	BuildingInstallation
IfcBeam	BuildingInstallation

IfcStair	BuildingInstallation
IfcFlowTerminal	BuildingInstallation
IfcFurnishingElement IfcBuildingElementProxy	BuildingFurniture

3.2.4 Implementation

As part of deliverable D3.2, a transformation was implemented.

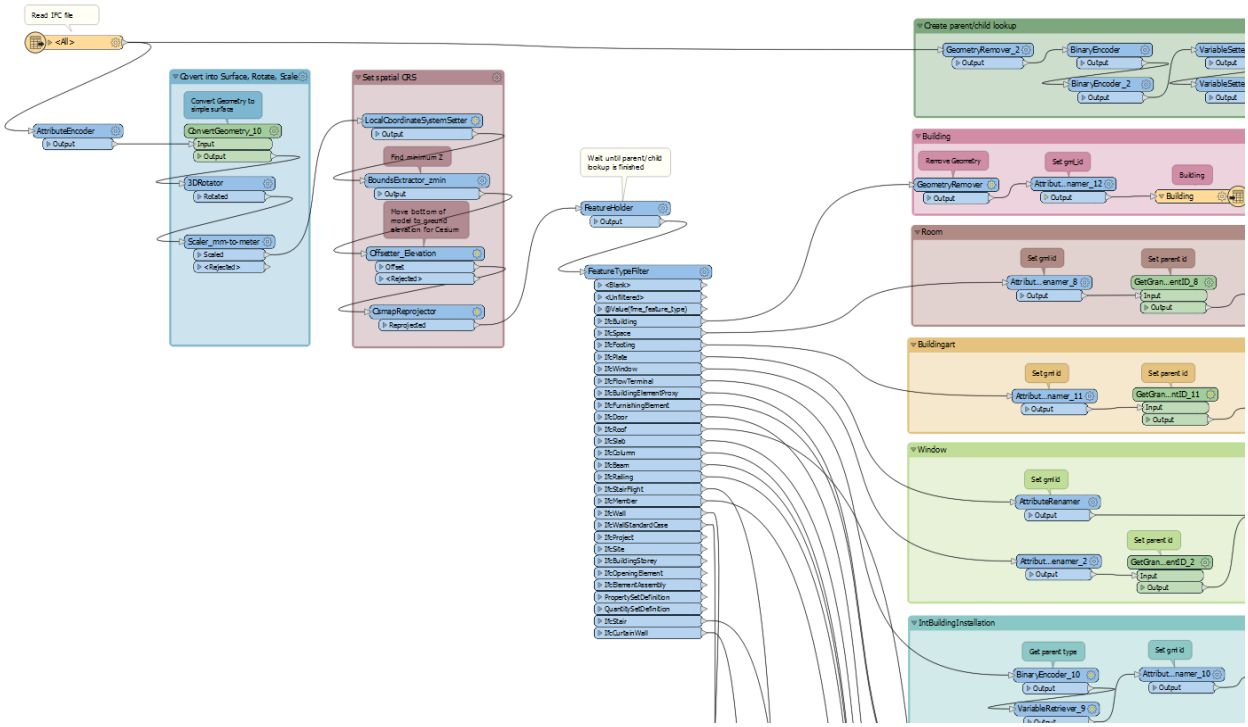


Figure 16 An implementation of the IFC-to-CityGML transformation

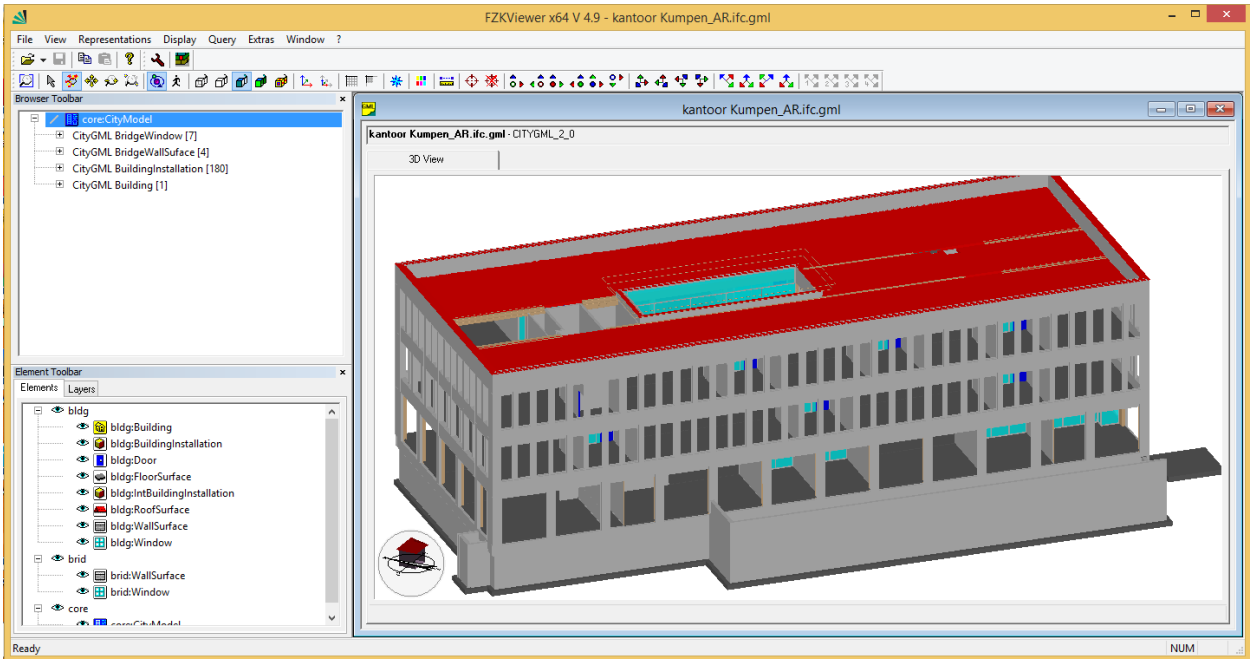


Figure 17 Conversion of the Kumpen Offce IFC BIM model (Willemen, 2018) into CityGML, viewed with FZKViewer

3.2.5 Future Improvements

We have identified the following improvements:

- **CityGML 3.0:** As CityGML 3.0 was not finalized yet at the time of writing this deliverable, the implementation was based on a draft version of the City GML 3.0 schemas. This XML Schema may be still subject to change.

3.3 IFC to 3D Tiles

This section describes how to convert an IFC BIM model into 3D Tiles that can be displayed in a Web application using CesiumJS.

3.3.1 Related requirements

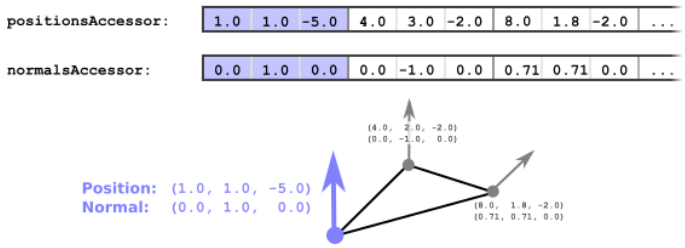
This translation relates to the following requirements described in deliverable D1.2 (BIMy consortium, 2019a):

- Req_Y1_057: Visualisation of a simplified 3D IFC Model for the fire brigade
- Req_Y1_051: Visualisation of public and private hydrants using a combination of GIS / BIM information
- Req_Y1_040: Visualisation of situational plan (BIM+GIS)
- Req_Y1_032: Visualize buildings and building parts

3.3.2 Transformation rules

A transformation from IFC to 3D Tiles encompasses among others the following transformations:

- **Geometry conversion**: 3D tiles use a mesh geometry defined in the glTF 2.0 specification (*GlTF - GL Transmission Format*, n.d.). Therefore, the complex Brep and CSG geometries of IFC have to be converted into mesh geometries, which basically consist of a lot of triangles.



- **Construct aggregates**: In IFC, some instances are composed of other instances. For example, curtainwalls contain plates and members; stairs contain railings, stair flights and slabs; and ramps contain slabs and ramps. These parts must be added to their parent feature when converting to CityGML.
- **Georeferencing**: 3D Tiles uses a right-handed Cartesian coordinate system; that is, the cross product of x and y yields z. 3D Tiles defines the z axis as up for local Cartesian coordinate systems; typically the same vertical reference system (VRS) as the terrain model must be chosen.
    - **Scale:** adjust the unit of measurement
    - **Rotate:** it may be needed to rotate the objects. This is particularly needed if the y-axis is not directed at the true north.
    - **Offset**: It may be needed to offset the x,y coordinates in the IFC file according to the required translation defined in the IFC file's IfcGeometricRepresentationContext WorldCoordinateSystem attribute. It may also be needed to adjust elevation, for instance, CesiumJS viewers often use an elevation above the WGS84 ellipsoid whereas IFC often uses an elevation above the sea level (e.g. Belgian TAW).
    - **Reprojection**: after applying the above-mentioned scaling, rotation, and offset, the coordinates of the IFC file may be in a geodetic coordinate reference system. 3D Tiles often require reprojected into a EPSG 4979, which is an earth-centered, earth-fixed (ECEF) reference frame.

3.3.3 Implementation

The above-mentioned translation was implemented using FME. Figure 18 depicts the FME workspace that was implemented.

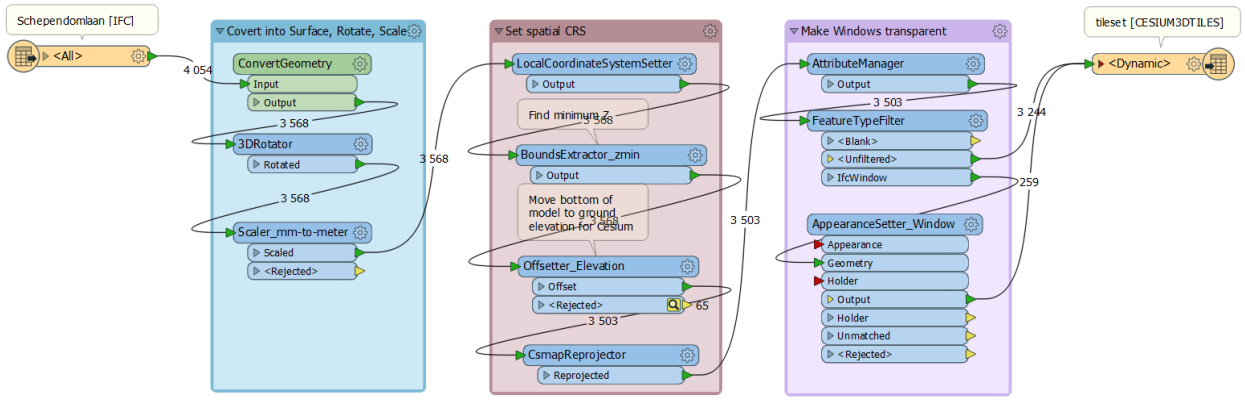


Figure 18 An implementation of the IFC-to-3DTiles conversion in FME

Figure 19 Screenshot of a simple Web application that visualizes 3D Tiles of the Schependomlaan IFC (openBIMstandards, 2014)

3.3.4 Future Improvements

The following further improvements can be made:

- **Implement aggregate construction**: the transformation rule for aggregate construction was not yet implemented.
- **Improve styling**: Several experiments were conducted to implement styling in the 3DTiles, this can be further improved.

3.3.5 Revit to 3D Tiles

In addition to a conversion from IFC to 3D tiles, a similar conversion from Revit to 3D tiles was implemented with FME. Figure 20 shows a subset of the FME workspace involving various custom

transformers for the conversion of Revit to 3DTiles directly without passing via IFC. Figure 21 depicts a visualization of the Kortrijk nursing home Revit file (ASSAR, 2019) using CesiumJS (Web Viewer)

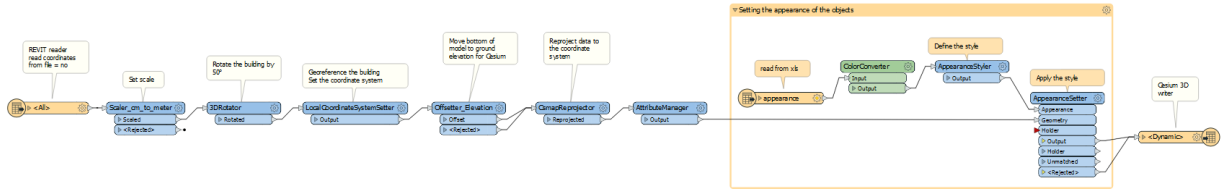


Figure 20 An implementation of Revit-to-3DTiles with FME

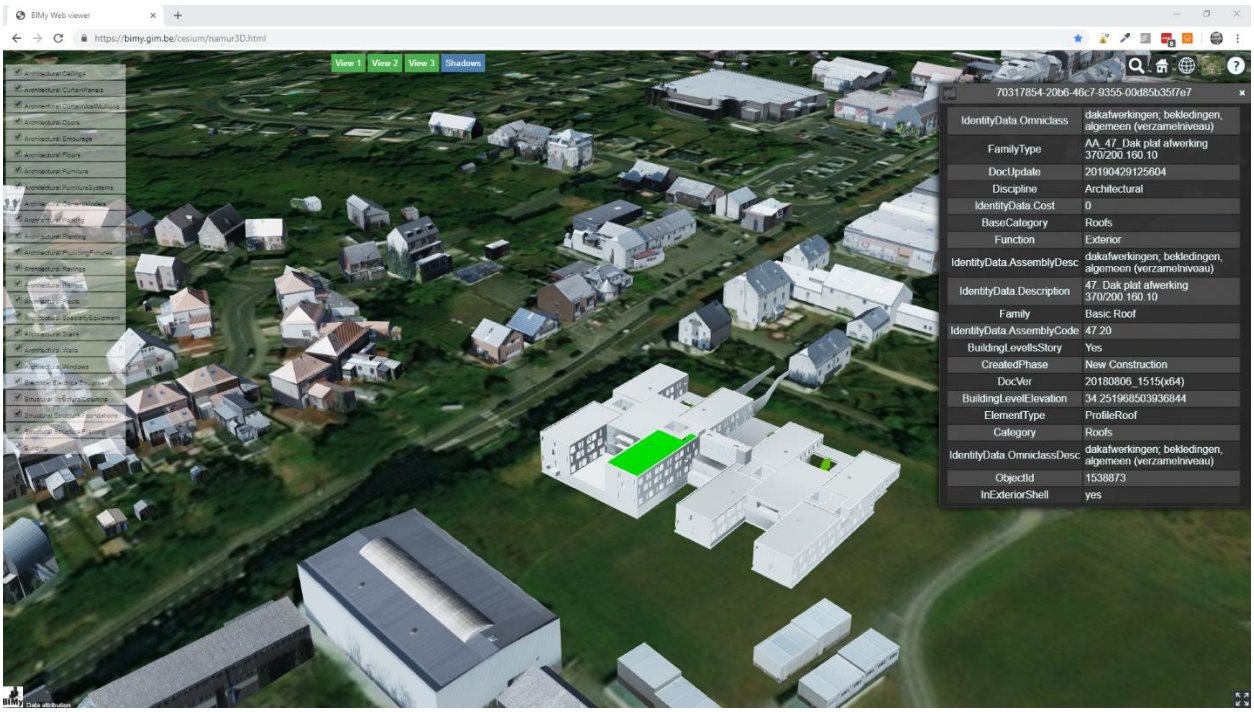


Figure 21 A visualization of the Kortrijk nursing home Revit file (ASSAR, 2019) using the BIM/GIS Integrated Web Viewer

4 GIS-to-BIM Mappings

This section formalizes the semantic and geometric transformations that are needed between the GIS and BIM data standards that are used in the BIMy platform. The starting point is the GIS data in CityGML, shape, GeoTIFF, GML or LAS data formats.

Different types of geospatial data that are relevant to the architects and construction companies as stakeholders of the Urban Context use case are:

- **Cadastral parcels**: multi-polygon geometries with attribute data typically provided in GIS formats as ESRI shape file (De-facto Industry standard) or alternatively in the Geography Markup Language (GML).
- **Utility networks**: to model utility networks in an uniform way, the Flemish government developed the IMKL (*Informatie Model Kabels En Leidingen V2.3*, 2017)information model basing itself on the standard that the EC developed for the modelling of utility infrastructure (*INSPIRE Data Specification on Utility and Government Services V3.0*, 2013) in the context of the INSPIRE framework directive. IMKL encoded in OGCs' Geography Markup Language is the selected data model and format to exchange information between the organization that own and operate subsoil utility network infrastructure and organization that need to conduct trenching operations. The *Kabel- En Leidinginformatieportaal (KLIP)*, n.d.) is a web platform where excavation and engineering companies can introduce requests for information (map requests) for a specific zone and utility companies that have infrastructure within this zone need to reply with the positional and attribute information of the infrastructure that they have at this location encoded in IMKL. The portal superimposes the data of the individual infrastructure owners and portrays them in accordance with the PMKL styling model (*Presentatiemodel Kabels En Leidingen V2.2*, n.d.). A similar model has been adopted by the Netherlands and it is expected that the other Belgian regions will adopt the same IMKL model. The same principles are obviously applicable on other EU Member states where the data is made available according to the base INSPIRE model for government and utility services.
- **Transport networks** and even better road surface geometries. Also here the INSPIRE initiative of the European Commission standardized the information model for Transport networks. The data model is specified in an Implementing rule and accompanying technical guidance documents (*D2.8.I.7Data Specification on Transport Networks – Technical Guidelines*, 2014). For the BIMy Urban context the road surface geometries that are made available as part of the "Grootschalige Basiskaart" or GRB and equivalents of the other regions that are made available as open data are used.
- **Buildings**: Also buildings is one of the core INSPIRE data themes. Here GIMs Belmap product was used.
- **Elevation data**: this concerns the topography of the terrain and is also referred to as Digital Terrain Models (DTM) not to be confused with Digital Surface Models (DSMs) which also contain the height of objects that are positioned on the ground like buildings. Digital Terrain models are typically distributed as raster data with a height value per grid cell in formats as GeoTIFF or JPEG2000. In the BIMy Urban context the digital terrain models that are made available as open data by the regional governments are used.

- **Ortho imagery**: this refers to typically airborne imagery that is orthorectifed (corrected for elevation differences) and made available in raster formats.
- **Mobile mapping data**: data that is typically acquired by carmounted 360° cameras and provide 360 degrees of the environment. These images can be draped on building facades but have the disadvantage of only representing the front facades.
- **Point cloud:** Point clouds collected from airborne or mobile mapping sensors provide the means to map the 3D geometry of objects. In the BIMy urban context open airborne LiDar data was used in combination with the Belmap footprints to reconstruct the 3D geometries of Buildings to a CityGML Level of detail 2.

4.1 2D GIS Vector data to IFC

Two-dimensional GIS vector datasets basically consist of point, linear and polygonal features (and multi-point, line and polygons) with associated attributes that are used to model the urban context. Several GIS formats are in use. They range from the proprietary ESRI shape file and geodatabase formats over open standards as OGCs Geography Markup Language and GeoJSON and can also be stored in geospatially aware databases as PostGIS or Oracle Spatial. The following datasets have been used in the BIMy Urban Context demo:

Type of data	Source	Format	Geometry
Buildings	Belmap	PostGIS	Polygon
Cadastral parcels	Open data	Shape	Polygon
Underground infrastructure	KLIP	IMKL – GML	Points, lines and polygons
Roads and road layout	Open data	Shape	Lines and polygons

4.1.1 Related requirements

This translation relates to the following requirements described in deliverable D1.2 (BIMy consortium, 2019b, p. 2):

- BE: Integrate urban context information into BIM models

4.1.2 Transformation rules

A transformation from GIS vector data formats into IFC encompasses among others the following transformations:

Transforming from tabular representation into a tree hierarchy: GIS vector data is typically stored in a tabular form, where each row in the dataset represents a different object and the columns represent the properties per object. In IFC on the other hand, data is organized in an object-based hierarchy. For example: A single GIS vector dataset containing buildings should be transformed into a hierarchy of IfcProject, IfcSite, IfcBuilding.

Create property sets: As described above, properties of GIS vector data are typically stored as columns in the dataset. When transforming this kind of data into IFC, these properties need to be converted into

propertysets. Properties that do not fit within the predefined properties of an IFC class can be added as user defined IfcPropertySets to extend what is foreseen in the IFC standard. For each set of user defined properties a IfcPropertySetDefinition should be created. The IfcPropertySetDefinition defines the schema of these properties.

Georeferencing: In many cases BIM models are created using a local reference point and cartesian coordinates, while GIS data is typically digitized using standardized geospatial coordinate systems. If the IFC data is intended to be used within a BIM tool together with a BIM model that uses a local reference system, the GIS data should be converted to the local reference system of the BIM model. This means that the GIS data will be rotated and offset to match the local reference system of the BIM model.

4.1.3 Implementation

The above-mentioned translation was implemented using FME. Figure 22 depicts a part of the FME workspace that was implemented to convert building information into IFC. Figure 23 shows the implementation to extend the IfcBuilding objects with additional IfcPropertySets.

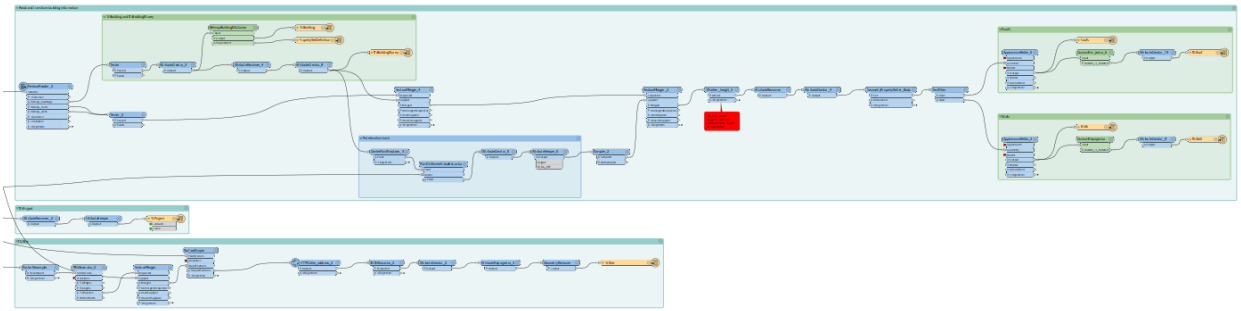


Figure 22 An implementation converting GIS vector data into IFC

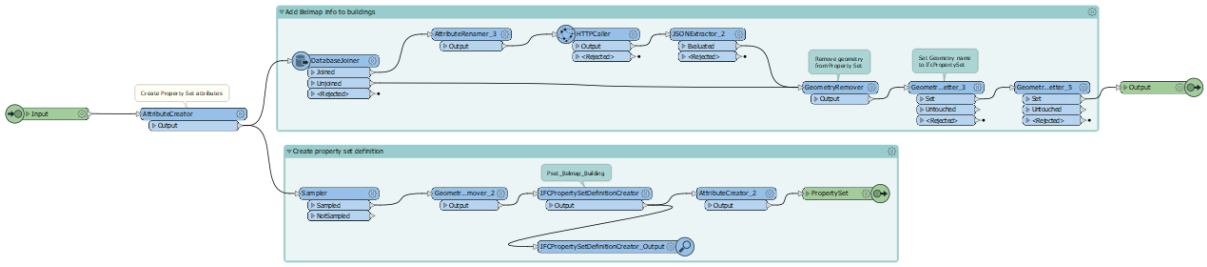


Figure 23 An implementation to add additional properties to IfcBuildings and to create the corresponding IfcPropertySetDefinition.

4.1.4 Known limitations

As demonstrated, the developed workflow can handle a wide variety of geospatial objects, data models and formats. One difficulty is to automatically transfer complex symbology as for instance the PMKL styling for underground infrastructure.

4.2 GIS raster data to IFC

Digital Elevation Model (GeoTiff) and orthoimagery (GeoTiff) to IFC

4.2.1 Related requirements

This translation relates to the following requirements described in deliverable D1.2 (BIMy consortium, 2019b, p. 2):

- BE: Integrate urban context information into BIM models
- The IFC4x1 Alignment standard provides a Terrain Surface object type.
- http://www.buildingsmart-tech.org/ifc/IFC4x1/RC2/html/link/terrain-surface.htm

4.2.2 Transformation rules

Geometric representation of tessellated surfaces using the IfcTesselatedItem which is a specialized class of the IfcGeometricRepresentationItem–The attribute Coordinates of the class IfcTesselatedFaceSet represents the point list of the surface–The attribute CoordIndex of the ifcTriangulatedFaceSet class represents the list of triangle. Semantic representation of the DTM using the class IfcGeographicElement which is subordinated to IfcSite

4.2.3 Implementation

By using a Digital Elevation Model we can transform the 2D GIS vector data, e.g. parcels, roads or waterways, into 3D objects that match the elevation of the terrain. This allows us to create a more accurate digital representation of an area. The usage of a DEM is included in the implementation depicted in Figure 22.

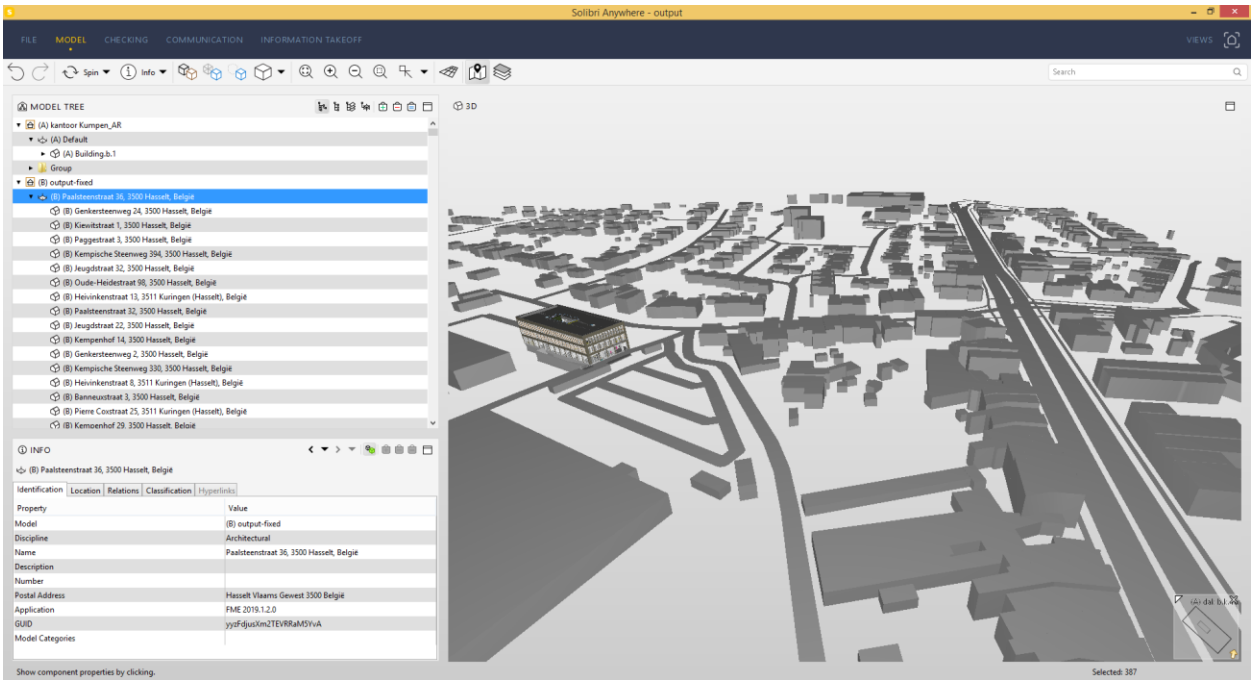


Figure 24 Screenshot of Solibri showing the Kumpen Office AR model and the Belmap-to-IFC output

4.2.4 Known limitations

No texture information in IFC, this may not be a problem, because texturing information is often added later in a the BIM modeling software or rendering engine. For example, in Revit, the use of RPC-based families allow to provide a simplified “cardboard” appearance, but when rendered, the view includes the right textures.

4.3 Point cloud to IFC or Revit

4.3.1 Related requirements

This translation relates to the following requirements described in deliverable D1.2 (BIMy consortium, 2019b, p. 2):

- BE: Integrate urban context information into BIM models
- The Architect can choose between GIS datasets with different LODs and versions. GIS Information is downloaded in a format which can be directly imported in BIM software (i.e.: IFC)

The idea here is not to integrate the raw point cloud data into BIM software, but to use the LiDAR data to reconstruct the building and other geometries and integrate this into the IFC model and Revit.

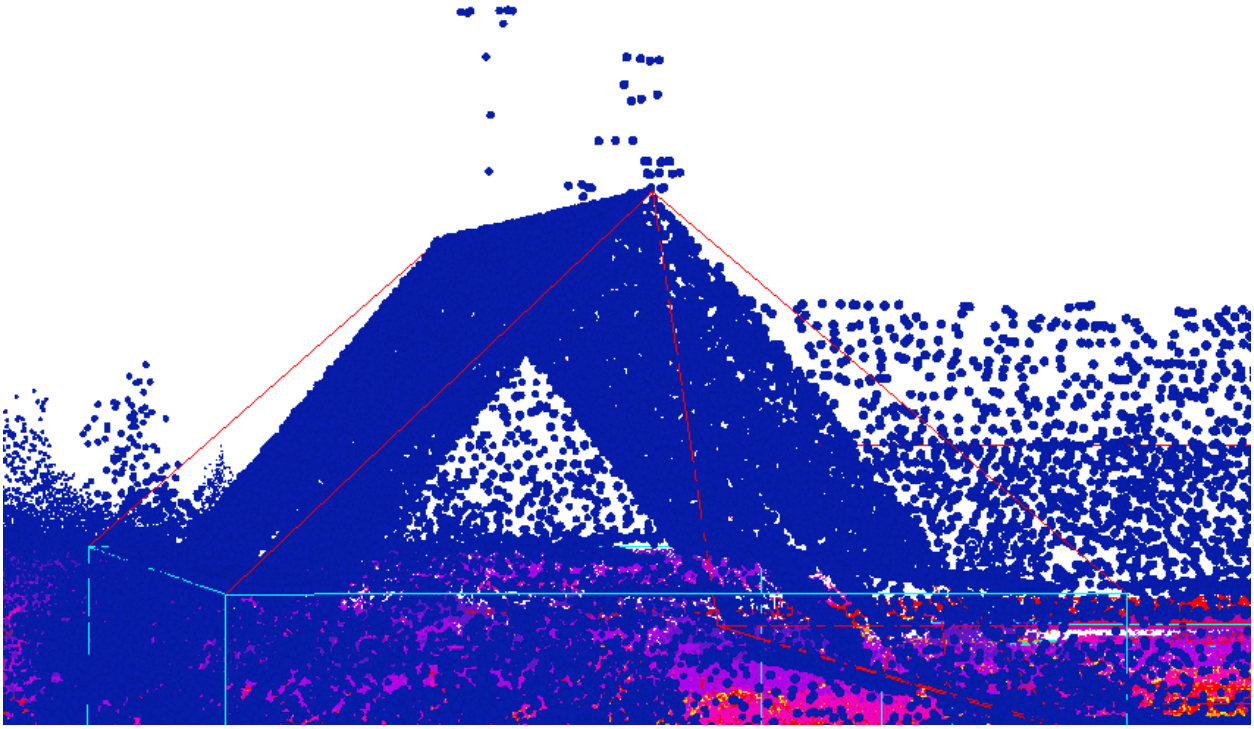


Figure 25 Example LiDAR point cloud and 3D Building reconstruction

4.3.2 Transformation Rules

Transform point cloud data into 2D or 3D GIS vector data: Since we do not want to import the raw point cloud data directly in a BIM tool, we first need to process the raw data and convert it into GIS vector data.
For buildings that meant reconstructing planar objects like walls and roofs from the point cloud data as described in section **Error! Reference source not found.**.
For trees the tree center point on the surface was extracted, as well as the tree height and the crown maximum diameter. Based on the position, height and crown diameter 3D representations of the trees in the area can be constructed. An example of this is shown in Figure 27.

Transform GIS vector data to IFC: Once the point cloud data is transformed into GIS vector data the transformation rules described in section 4.1.2 can be used to transform the data into IFC.

4.3.3 Implementation

A thorough evaluation of different software packages was performed in terms of

- the quality of the automatic reconstruction,
- the fact whether or not the Belmap footprint could be introduced to constrain the building reconstruction
- support for varying input data sources (classified versus non-classified point clouds) and support for processed DSMs,

- possibility to integrate other geometries like cutlines (to separate buildings in a building block and) roof ridges to further constrain the roof reconstruction
- the factor whether the output consist of a mesh structure or real 3D solid objects.
- The associated costs.

Pilot zones in the Flemish region and Gembloux in the Walloon region were selected, this to investigate both the effect of the density of buildings (rural-residential versus City) and the point density of the point clouds.

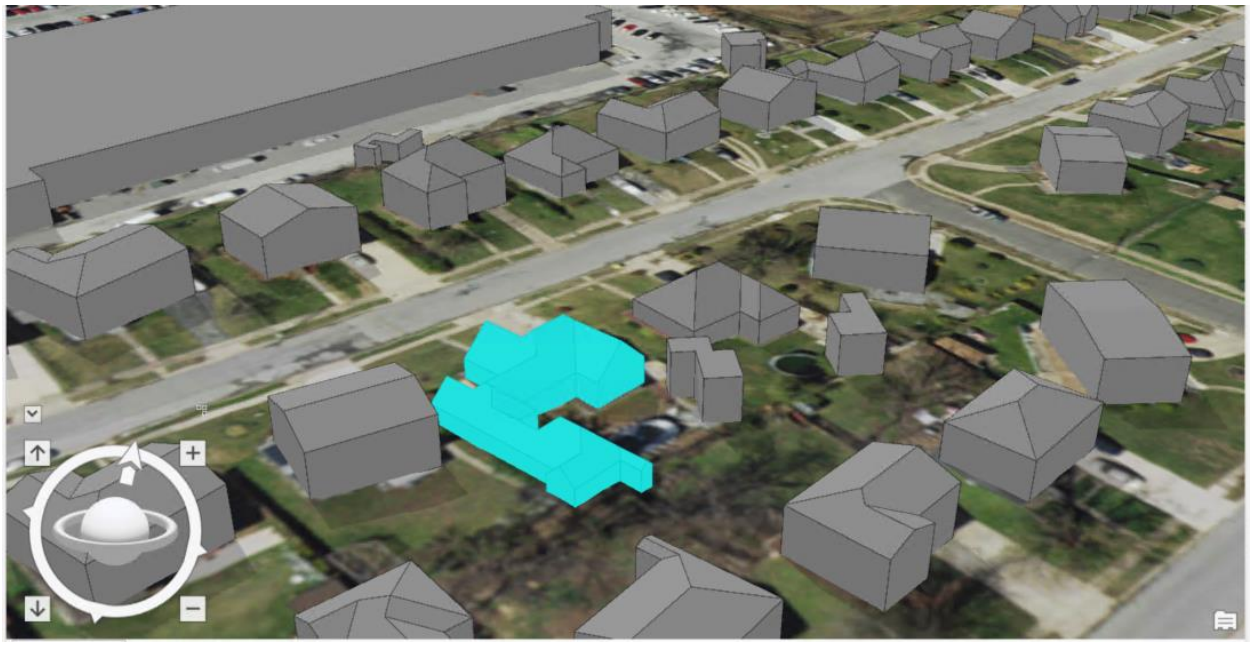


Figure 26Example buildings reconstructed

Also tree objects were extracted from the classified point cloud using an adapted processing chain that extracted the height of the tree and the maximum crown diameter.

Figure 27 Extraction of trees from LiDAR point clouds

4.3.4 Future improvements

The following future improvements can be noted with respect to the LiDAR based building reconstruction:

- Scalability: the resulting workflow is complex and needs to work on a huge amount of data exceeding 50TB (uncompressed). Further optimisations to the processing chain are required to be able to generate this for the entire country.
- Optimum to be found in the level of detail of the building features that are reconstructed and artefacts that are created because of overhanging trees or adjacent building structures.

4.4 IFC/REVIT to Video/Gaming/3D Engine

Finally it is the idea to also export the integrated BIM/GIS data model to gaming engines like Unity and Unreal.

4.4.1 Related requirements

This translation relates to the following requirements described in deliverable D1.2 (BIMy consortium, 2019b, p. 2):

- BE: Integrate urban context information into BIM models
- The Architect creates a video or VR of the project

4.4.2 Transformation rules

Georeferencing: A gaming engine is - unlike GIS viewers - typically not built to be able to deal with data in different coordinate reference systems. Because we want to include data from different sources (e.g. a 3D Revit model and GIS Vector data) we should make sure all data is converted into the same cartesian coordinate reference system.

Transforming to compatible file format: Once all data is transformed to the same coordinate system, it needs to be transformed to a file format that can be read by the engine of choice. This could be e.g. OBJ, AutoDesk FilmBox or Unreal Datasmith.

Importing into gaming engine: The data needs to be imported into the gaming engine and added to a virtual scene or world. To be able to walk around a player needs to be added. To get interaction between the player and the virtual world (e.g. to be able to walk up stairs) a collision model can be added to the correct objects.

Compiling the game: When the virtual world is set up, it needs to be exported into a standalone executable file.

4.4.3 Implementation

Figure 28 depicts the implementation of the process to transform IFC data into the AutoDesk FilmBox format.

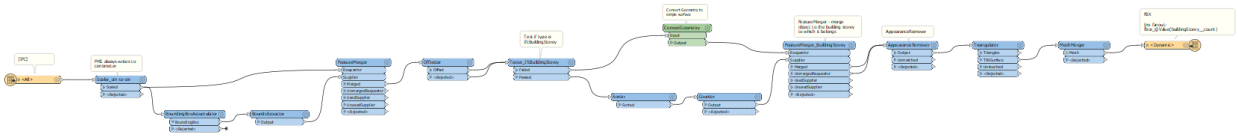


Figure 28 An implementation of a transformation from IFC to FilmBox

4.4.4 Future improvements

The existing implementation proves that the BIM/GIS integrated model can be exported to Gaming engines like Unreal and Unity. Further improvements can be done to realistic model the texture of the facades using segmented information from for instance oblique imagery.

5 Linked BIM/GIS data: BIMy Data Model

The BIMy data model represents the minimal BIM and GIS objects that should be managed on the BIMy platform. The BIMy Data Model is represented as a conceptual model, an XML Schema (GML Application Profile), and an RDF Schema.

5.1 Rationale

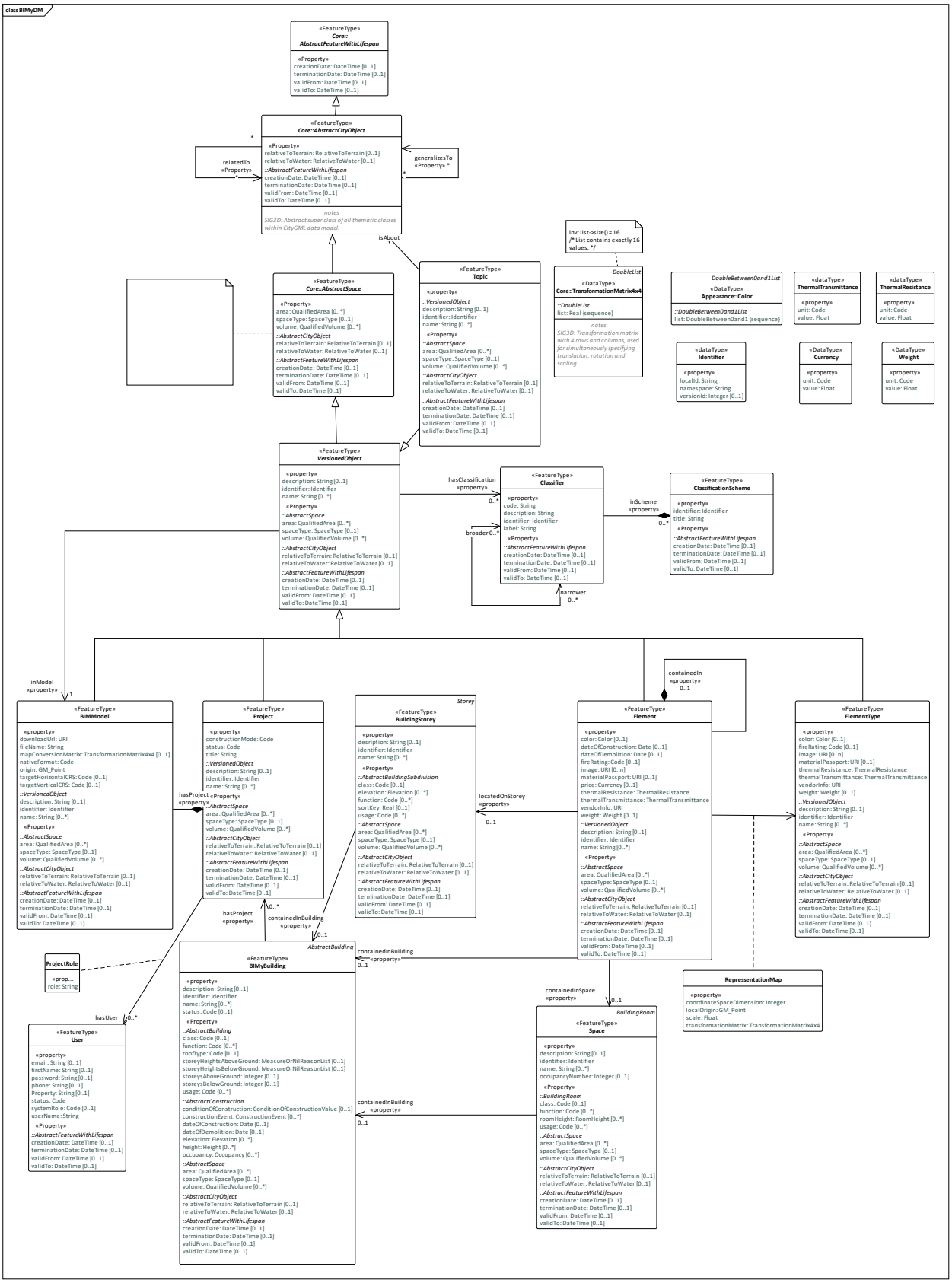
Via the BIMy API users can upload BIM models on the BIMy platform in one of the supported upload formats, such as IFC2x3, IFC4, or Revit. The BIMy platform will store the BIM model as a file, but additionally it will read a filtered set of relevant objects from the BIM model, convert these objects into a simplified, yet fully georeferenced, representation - this is the BIMy data model - and store this data in a datastore on the BIMy platform. The data store will then allow users to search, visualise, annotate, raise issues on, etc. these elements. Because the platform knows for each element in which BIM model(s) it occurs, the user can also download the corresponding BIM model (either as it was uploaded or converted into another format) and use a dedicated BIM tool for further analysis.

5.2 Conceptual model: BIMy Data Model

The core of the BIMy Data Model supports the following key requirements:

- **BIMModel**: A BIMModel is defined as a snapshot of a BIM model represented in a native format (e.g. Revit, IFC2x3, etc.). The BIMy data model makes it possible to find out in which BIMModel a VersionedObject occurs. The BIMy Data Model therefore does not need to have the same level of detail (level of geometry, level of information need) as the linked BIMModel.
- **Identifier**: The data type Identifier consists of a namespace (a URI), a localId, and an optional versionId. The concatenation of the {namespace}{localId} attributes makes up a (preferably HTTP) URI. This enables a "Linked Construction Data" approach where each Object has a Web identifier (HTTP URI). The BIMy ReST API will resolve this URI to a CityGML or JSON document describing this object. The combination of {namespace}{localId} (without versionId) is also expected to be a stable object identifier that can be used to refer to the object during its lifecycle.
- **Object-level versioning**: In CityGML 3.0 every FeatureType is a subclass of the abstract class AbstractFeatureWithLifespan. The validFrom and validTo dates indicate the validity of the object in the real world. The creationDate and terminationDate are the dates the object was created.
- **Basic object properties**: each object has very basic properties such as name, description, etc. that are inherited from GML.
- **Project, Building, BuildingStorey, Space**: the model allows representing key BIM/CityGML entities. Note that Building has a building footprint property (multipolygon), which is a typical GIS concept.
- **Element**: all key IFC building elements are modeled as Element on the BIMy platform. An Element may be classified by one or more Classifiers.

- **Support multiple levels of geometry**: An element must be modeled as a (simplified) GML MultiSurface geometry. For the moment, only LoD 1 and LoD 2 are considered. Also, only simple surface geometries are in scope of the BIMy platform, as the main requirement is visualisation of these geometries.
- **ElementType**: An element may also be linked to an ElementType. The IFC standards have a similar concept.
- **Support shared representations**: Rather than instantiating each geometry, the BIMy data model can support common shape representations with the RepresentationMap concept, greatly reducing the amount of storage required to store frequently occurring geometries (e.g. a water hydrant shape). The IFC standard has a similar concept (IfcRepresentationMap).


class BIMyDM
«FeatureType» Core:: AbstractFeatureWithLifespan
«Property» creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
«FeatureType» Core::AbstractCityObject
«Property» relativeToTerrain: RelativeToTerrain [0..1] relativeToWater: RelativeToWater [0..1]
::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
notes
SIG3D: Abstract super class of all thematic classes within CityGML data model.
relatedTo «Property»
generalizesTo «Property» *
isAbout
«FeatureType» Core::AbstractSpace
«Property» area: QualifiedArea [0..*] spaceType: SpaceType [0..1] volume: QualifiedVolume [0..*]
::AbstractCityObject relativeToTerrain: RelativeToTerrain [0..1] relativeToWater: RelativeToWater [0..1]
::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
«FeatureType» Topic
«property» ::VersionedObject description: String [0..1] identifier: Identifier name: String [0..*]
«Property» ::AbstractSpace area: QualifiedArea [0..*] spaceType: SpaceType [0..1] volume: QualifiedVolume [0..*]
::AbstractCityObject relativeToTerrain: RelativeToTerrain [0..1] relativeToWater: RelativeToWater [0..1]
::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
inv: list->size() = 16
/* List contains exactly 16 values. */
DoubleList
«DataType» Core::TransformationMatrix4x4
::DoubleList list: Real [sequence]
notes
SIG3D: Transformation matrix with 4 rows and columns, used for simultaneously specifying translation, rotation and scaling.
DoubleBetween0and1List
«DataType» Appearance::Color
::DoubleBetween0and1List list: DoubleBetween0and1 [sequence]
«dataType» ThermalTransmittance
«property» unit: Code value: Float
«dataType» ThermalResistance
«property» unit: Code value: Float
«dataType» Identifier
«property» localId: String namespace: String versionId: Integer [0..1]
«DataType» Currency
«property» unit: Code value: Float
«DataType» Weight
«property» unit: Code value: Float
«FeatureType» VersionedObject
«property» description: String [0..1] identifier: Identifier name: String [0..*]
«Property» ::AbstractSpace area: QualifiedArea [0..*] spaceType: SpaceType [0..1] volume: QualifiedVolume [0..*]
::AbstractCityObject relativeToTerrain: RelativeToTerrain [0..1] relativeToWater: RelativeToWater [0..1]
::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
hasClassification «property» 0..*
«FeatureType» Classifier
«property» code: String description: String identifier: Identifier label: String
«Property» ::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
broader 0..*
narrower 0..*
inScheme «property» 0..*
«FeatureType» ClassificationScheme
«property» identifier: Identifier title: String
«Property» ::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
inModel «property» 1
«FeatureType» BIMModel
«property» downloadUrl: URI fileName: String mapConversionMatrix: TransformationMatrix4x4 [0..1] nativeFormat: Code origin: GM_Point targetHorizontalCRS: Code [0..1] targetVerticalCRS: Code [0..1]
::VersionedObject description: String [0..1] identifier: Identifier name: String [0..*]
«Property» ::AbstractSpace area: QualifiedArea [0..*] spaceType: SpaceType [0..1] volume: QualifiedVolume [0..*]
::AbstractCityObject relativeToTerrain: RelativeToTerrain [0..1] relativeToWater: RelativeToWater [0..1]
::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
«FeatureType» Project
«property» constructionMode: Code status: Code title: String
::VersionedObject description: String [0..1] identifier: Identifier name: String [0..*]
hasProject «property» 0..*
«Property» ::AbstractSpace area: QualifiedArea [0..*] spaceType: SpaceType [0..1] volume: QualifiedVolume [0..*]
::AbstractCityObject relativeToTerrain: RelativeToTerrain [0..1] relativeToWater: RelativeToWater [0..1]
::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
Storey
«FeatureType» BuildingStorey
«property» description: String [0..1] identifier: Identifier name: String [0..*]
«Property» ::AbstractBuildingSubdivision class: Code [0..1] elevation: Elevation [0..*] function: Code [0..*] sortKey: Real [0..1] usage: Code [0..*]
::AbstractSpace area: QualifiedArea [0..*] spaceType: SpaceType [0..1] volume: QualifiedVolume [0..*]
::AbstractCityObject relativeToTerrain: RelativeToTerrain [0..1] relativeToWater: RelativeToWater [0..1]
::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
locatedOnStorey «property» 0..1
containedIn «property» 0..1
«FeatureType» Element
«property» color: Color [0..1] dateOfConstruction: Date [0..1] dateOfDemolition: Date [0..1] fireRating: Code [0..1] image: URI [0..n] materialPassport: URI [0..1] price: Currency [0..1] thermalResistance: ThermalResistance thermalTransmittance: ThermalTransmittance vendorInfo: URI weight: Weight [0..1]
::VersionedObject description: String [0..1] identifier: Identifier name: String [0..*]
«Property» ::AbstractSpace area: QualifiedArea [0..*] spaceType: SpaceType [0..1] volume: QualifiedVolume [0..*]
::AbstractCityObject relativeToTerrain: RelativeToTerrain [0..1] relativeToWater: RelativeToWater [0..1]
::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
«FeatureType» ElementType
«property» color: Color [0..1] fireRating: Code [0..1] image: URI [0..n] materialPassport: URI [0..1] thermalResistance: ThermalResistance thermalTransmittance: ThermalTransmittance vendorInfo: URI weight: Weight [0..1]
::VersionedObject description: String [0..1] identifier: Identifier name: String [0..*]
«Property» ::AbstractSpace area: QualifiedArea [0..*] spaceType: SpaceType [0..1] volume: QualifiedVolume [0..*]
::AbstractCityObject relativeToTerrain: RelativeToTerrain [0..1] relativeToWater: RelativeToWater [0..1]
::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
hasProject «property» 0..*
containedInBuilding «property» 0..1
AbstractBuilding
«FeatureType» BIMyBuilding
«property» description: String [0..1] identifier: Identifier name: String [0..*] status: Code [0..1]
«Property» ::AbstractBuilding class: Code [0..1] function: Code [0..*] roofType: Code [0..1] storeyHeightsAboveGround: MeasureOrNilReasonList [0..1] storeyHeightsBelowGround: MeasureOrNilReasonList [0..1] storeysAboveGround: Integer [0..1] storeysBelowGround: Integer [0..1] usage: Code [0..*]
::AbstractConstruction conditionOfConstruction: ConditionOfConstructionValue [0..1] constructionEvent: ConstructionEvent [0..*] dateOfConstruction: Date [0..1] dateOfDemolition: Date [0..1] elevation: Elevation [0..*] height: Height [0..*] occupancy: Occupancy [0..*]
::AbstractSpace area: QualifiedArea [0..*] spaceType: SpaceType [0..1] volume: QualifiedVolume [0..*]
::AbstractCityObject relativeToTerrain: RelativeToTerrain [0..1] relativeToWater: RelativeToWater [0..1]
::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
containedInBuilding «property» 0..1
ProjectRole
«prop... role: String
hasUser 0..*
«FeatureType» User
«property» email: String [0..1] firstName: String [0..1] password: String [0..1] phone: String [0..1] Property: String [0..1] status: Code systemRole: Code [0..1] userName: String
«Property» ::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
containedInSpace «property» 0..1
BuildingRoom
«FeatureType» Space
«property» description: String [0..1] identifier: Identifier name: String [0..*] occupancyNumber: Integer [0..1]
«Property» ::BuildingRoom class: Code [0..1] function: Code [0..*] roomHeight: RoomHeight [0..*] usage: Code [0..*]
::AbstractSpace area: QualifiedArea [0..*] spaceType: SpaceType [0..1] volume: QualifiedVolume [0..*]
::AbstractCityObject relativeToTerrain: RelativeToTerrain [0..1] relativeToWater: RelativeToWater [0..1]
::AbstractFeatureWithLifespan creationDate: DateTime [0..1] terminationDate: DateTime [0..1] validFrom: DateTime [0..1] validTo: DateTime [0..1]
RepresentationMap
«property» coordinateSpaceDimension: Integer localOrigin: GM_Point scale: Float transformationMatrix: TransformationMatrix4x4

5.3 XML Schema encoding (CityGML ADE)

The bimy.xsd XML Schema is an Application Domain Extension (ADE) of the CityGML 3.0 GML Application Schema.

5.4 Linked Data Vocabulary

The bimy.ttl vocabulary is an encoding of the BIMy Conceptual Data model based on BOT and SKOS.

Bibliography

Arroyo Ohori, K., Biljecki, F., Diakité, A., Krijnen, T., Ledoux, H., & Stoter, J. (2017). Towards an integration of GIS and BIM data: What are the geometric and topologial issues? *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *IV-4/W5*, 1–8. https://doi.org/10.5194/isprs-annals-IV-4-W5-1-2017

ASSAR. (2019). *BIM model of a nursing home in Kortrijk*.

BIMy consortium. (2017). *BIMy—BIM in the City—ITEA3 Full Project Proposal*.

BIMy consortium. (2019a). *BIMy—BIM in the City—D1.2 Platform applications and requirements*.

BIMy consortium. (2019b). *BIMy—BIM in the City—D2.2 Integrated BIM/GIS model*.

Bonduel, M., Oraskari, J., Pauwels, P., Vergauwen, M., & Klein, R. (2018). The IFC to linked building data converter: Current status. *Proceedings of the 6th Linked Data in Architecture and Construction Workshop*, *2159*, 34–43. http://ceur-ws.org/Vol-2159/04paper.pdf

Chen, Y., Shooraj, E., Rajabifard, A., & Sabri, S. (2018). From IFC to 3D Tiles: An Integrated Open-Source Solution for Visualising BIMs on Cesium. *ISPRS International Journal of Geo-Information*, *7*(10). https://doi.org/10.3390/ijgi7100393

Clemen, C., & Görne, H. (2019). Level of Georeferencing (LoGeoRef) using IFC for BIM. *Journal of Geodesy, Cartography and Cadastre*, *10*.

D2.8.I.7Data Specification on Transport Networks – Technical Guidelines. (2014).

GlTF - GL Transmission Format. (n.d.).

Holten Rasmussen, M. (2018, June 19). *Ontology for Property Management*. LDAC2018 - 6th Linked Data in Architecture and Construction Workshop (19 - 21 June 2018), London. http://linkedbuildingdata.net/ldac2018/files/Presentations/20180619_OPM_madsholtenrasmussen.pdf

IfcOpenShell. (2019). *IfcOpenShell*. http://ifcopenshell.org/

Informatie Model Kabels en Leidingen V2.3. (2017). https://overheid.vlaanderen.be/help/klipimkl-formaat/imkl-23

i-SCOPE Project. (2013). *CityGML - Time Dependent Variables ADE*. http://www.citygmlwiki.org/index.php?title=Time_Dependent_Variables_ADE

ISO. (2018). *ISO 10303-42:2018 Industrial automation systems and integration—Product data representation and exchange—Part 42: Integrated generic resource: Geometric and topological representation*. ISO. https://www.iso.org/standard/76128.html

Lamonarch, D. (2019). *Dynamo Elk plugin*. https://dynamonodes.com/tag/elk/

Liangliang Nan, Peter Wonka. (n.d.). PolyFit: Polygonal Surface Reconstruction from Point Clouds. *ICCV 2017*.

Nagel C, Stadler, A, & Kolbe T. (2009). *Conceptual Requirements for the Automatic Reconstruction of Building Information Models from Uninterpreted 3D Models*. Academic Track of Geoweb 2009 Conference, Vancouver.

Noardo, F., Arroyo Ohori, K., Biljecki, F., Krijnen, T., Ellul, C., Harrie, L., & Stoter, J. (2019). GeoBIM Benchmark 2019: Design and initial results. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-2/W13*, 1339–1346. https://doi.org/10.5194/isprs-archives-XLII-2-W13-1339-2019

OGC. (2012). *OGC City Geography Markup Language (CityGML) Encoding Standard, version 2.0*. https://www.opengeospatial.org/standards/citygml

OGC. (2020). *OGC CityGML, version 3.0 - request for public comments - https://www.ogc.org/standards/requests/219*

OGC. (2019b). *OGC 3D Tiles Specification*. OGC. http://www.opengeospatial.org/standards/3DTiles

openBIMstandards. (2014). *Dataset Schependomlaan*. openBIMstandards. https://github.com/openBIMstandards/DataSetSchependomlaan

Poulain, P. (2018). *Community Experiments with Time-Dynamic 3D Tiles*. https://cesium.com/blog/2018/11/08/weather-prediction-data-time-series-and-3d-tiles/

Terkaj, W., Schneider, G. F., & Pauwels, P. (2017). Reusing domain ontologies in linked building data: The case of building automation and control. *Proceedings of the Joint Ontology Workshops 2017 Episode 3: The Tyrolean Autumn of Ontology*, *2050*, 12.

W3C. (2019). *Building Topology Ontology*. W3C. https://w3c-lbd-cg.github.io/bot/

Willemen. (2018). *BIM model of the Kumpen Office in Hasselt*.