Change toolkit for digital building permit

Deliverable number	D3.1
Deliverable name	Geo to BIM tool/procedure
Work package number	WP3 GeoBIM
Deliverable leader	RDF Ltd.
Dissemination Level	Public

Status	Final
Version Number	V1.0
Due date	M25
Submission date	30-10-2024

Project no. 101058559

Start date of project: 1 October 2022

Duration: 36 months

File name: CHEK_101058559_D3.1-Geo to BIM tool procedure_V1.0_Final

Funded by the European Union

This project has received funding from the European Union under the Horizon Europe Research & Innovation Programme 2021-2027 (grant agreement no. 101058559).

Authors and contributors

Author	Organisation	E-mail
Peter Bonsma	RDF	peter.bonsma@rdf.bg
Tzvetelina Zayakova	RDF	
Svilen Varbanov	RDF	

Quality control

Author	Organisation	Role	Date
Iveta Bonsma	RDF	WP leader	23/10/2024
Ken Arroyo Ohori	TUD	Reviewer	24/10/2024
Jantien Stoter	TUD	Coordinator	25/10/2024

Document history

Release	Description	Date	Author
V0.1	Initial version	01/10/2024	Peter Bonsma
V0.2	Draft release version	14/10/2024	Peter Bonsma / Svilen Varbanov
V0.3	Draft release version	16/10/2024	Peter Bonsma / Svilen Varbanov
V1.0	Reviewed by TUD	28/10/2024	Peter Bonsma / Ken Arroyo Ohori

Contents

1. Executive Summary

This document describes the architecture and components developed to enable a Geo to BIM conversion tool / procedure. The core development exists of two parts:

- GIS engine supporting open standards GML, CityGML, CityJSON, LandXML and InfraGML with support for profiles and ADEs (Application Domain Extensions) and geometrical representation;
- Geo to BIM converter, working for any valid GML, CityGML or CityJSON input file and generating a 100% valid IFC 4 ADD2 TC1 file.

1.1 GIS library

The developed GIS library is an extension of the already existing Geometry Modeling Kernel (http://www.geometrykernel.com/ from RDF) allowing 3D representation of any GEOM compatible ontology and a public API to enable read / write access to all instances and classes on the related Semantic Web content.
The work on creating this GIS library focused mainly on supporting the open standard GIS formats, automatically finding their related content and schemas (locally automatically downloaded XSD referenced files) and conversion of the content into an RDF / RDFS / OWL based representation. As the Semantic Web representation is using the GEOM ontology and conversion is complete, all data can be accessed as well as visualized in 2D and / or 3D.

1.2 Geo to BIM converter

Based on the developed GIS library and the existing BIM library (IFC Engine Library, i.e. http://www.ifcengine.com/ from RDF) the Geo to BIM converter is developed. This is an open-source development based on commercial products; however, using public / open ISO defined API's.
The converter allows any GML, CityGML (1.0, [08-007r1], 2.0 [12-019], 3.0 [20-010 / 21-006r2]) or CityJSON (1.0.0, 1.0.1, 1.0.2, 1.0.3, 1.1.0, 1.1.1, 1.1.2, 1.1.3, 2.0.0, 2.0.1) file (any version) and any ADE or profile used. The generated output is always an IFC4 ADD2 TC1 file (i.e. ISO 16739-1:2018) and the mapping can be configured for attribute to property-set + property mapping and (missing) material mapping based on CityGML classification.
The converter is available via the following components:

- https://github.com/peterrdf/gml2ifc open-source Geo to BIM converter (public GitHub page)
- https://rdf.bg/CHEK/gml2ifc.html Web Assembly based Online Service
- https://github.com/peterrdf/stepengineexamples/tree/main/bin/32bit/WASM distributable package

2. Introduction

The development of the GIS 2 BIM converter (Chapter 4) is applied using the existing libraries from RDF Ltd.At the start of the project there was no GIS library available. The base existing components were a Geometry Modeling Kernel and an IFC Engine (BIM supporting library).This meant a GIS library had to be developed based on the Geometry Modeling Kernel. Later on, the converter was developed based on both the new GIS library and existing BIM library (Chapter 3).

3. GIS library

The GIS library is developed as an extension of the existing Geometry Modeling Kernel. The Geometry Modeling Kernel now allows visualization and full API access of any Semantic Web content valid against the CHEK ontology. This meant that an RDF / RDFS / OWL representation of the GIS content was required.

More specifically, the development of a CityRDF representation of CityGML or CityJSON structured content is a very complex question in itself. Together with partners within the ACCORD project and under lead of partner OGC, the definition of a CityRDF is further developed to become an official serialization of the CityGML data model.

3.1 CityRDF

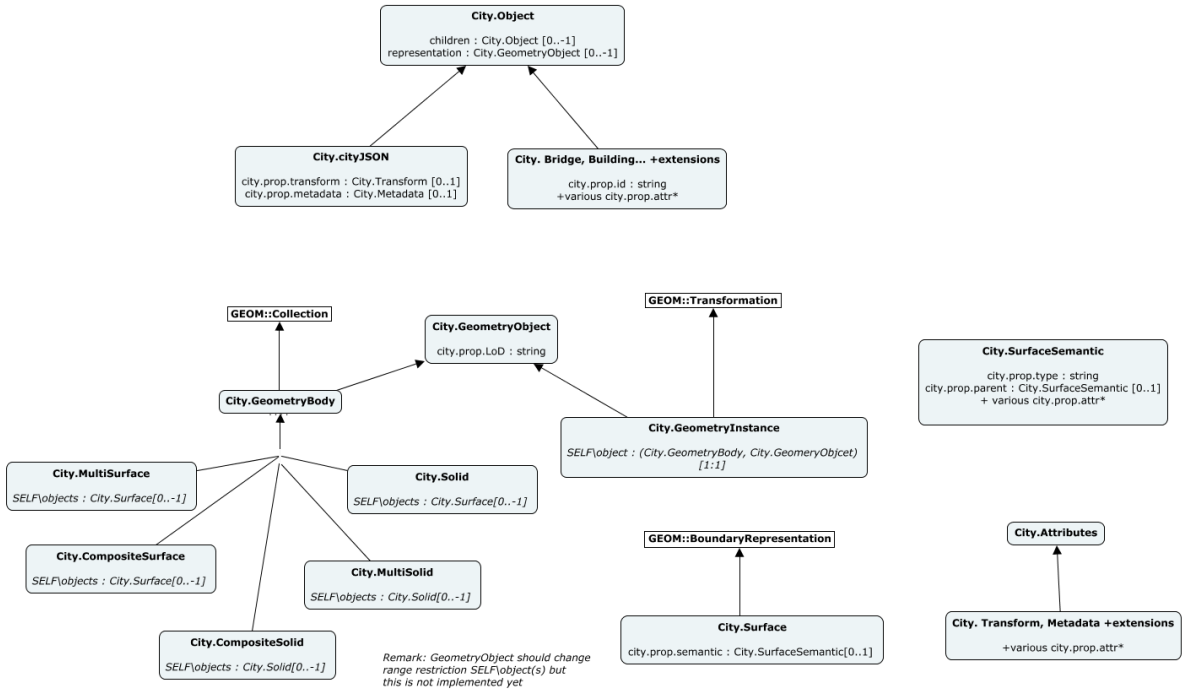


Figure 1 CityRDF Ontology

Within CHEK and specifically within the GIS library, there was no time to wait for this official development (https://github.com/ogcincubator/cityrdf) and an initial RDF / RDFS / OWL ontology was created for the CityGML data model. In this conversion the focus was on a 1-to-1 mapping of knowledge. This not only allows support for the CityGML data model, but also for so-called ADEs (Application Domain Extensions). A similar solution has been developed for CityJSON; while CityJSON as a serialization setup is very similar to the CityGML serialization setup, it has some distinct differences even though both are defined on the same CityGML data model. An exception in the conversion to RDF / RDFS / OWL was made for the geometrical representations. For geometrical representations, the GEOM ontology is used as a base, here not only basic geometrical representations had to be mapped but also materialization and support for repeated objects without losing the underlying semantics of repetition.

In a later stage it became clear not all content would be available as CityGML within CHEK; especially in the early stages much of the knowledge was available as GML only. The final version of the developed GIS engine supports therefore GML, CityGML and CityJSON; in time all recent versions of these open standards will be supported. Also, ADEs (Application Domain Extensions) are supported although this was eventually not necessary as CHEK CityGML was defined on profiles rather than ADEs. Additionally support for open standards LandXML and InfraGML has been developed for the GIS engine. Compared to the original task, i.e. support for CityGML and CityJSON, the open standards GML, LandXML and InfraGML were also supported; GML as it is the base for many of the OGC GIS related standards; LandXML and InfraGML as these standards are very close to the infrastructure extension as can be found in the most recent IFC standard; here InfraGML is the renewed OGC variant of the non-OGC open standard LandXML (therefore initially in Task 3.1 these seemed to be a logical choice). Although from tooling perspective both landXML and InfraGML support has not been used in CHEK, the GML support has proven to be essential as conversion from GML was in several cases early in the design process the only available solution to create BIM content from existing GIS content.

In a simplified view the different supported open-standards (GML, CityGML, CityJSON, LandXML and InfraGML) are converted towards an ontology that is created on-the-fly (to allow support for ADEs) partly integrated with the GEOM ontology representing the underlying geometry contained in the input. This allows the Geometry Modeling Kernel to directly visualize the converted geometry. An existing application, i.e. the 3DEditor, has been extended to visualize all supported open formats with all (non-geometrical and geometrical) semantic knowledge as well as the 2D / 3D renderings of the complete data set.

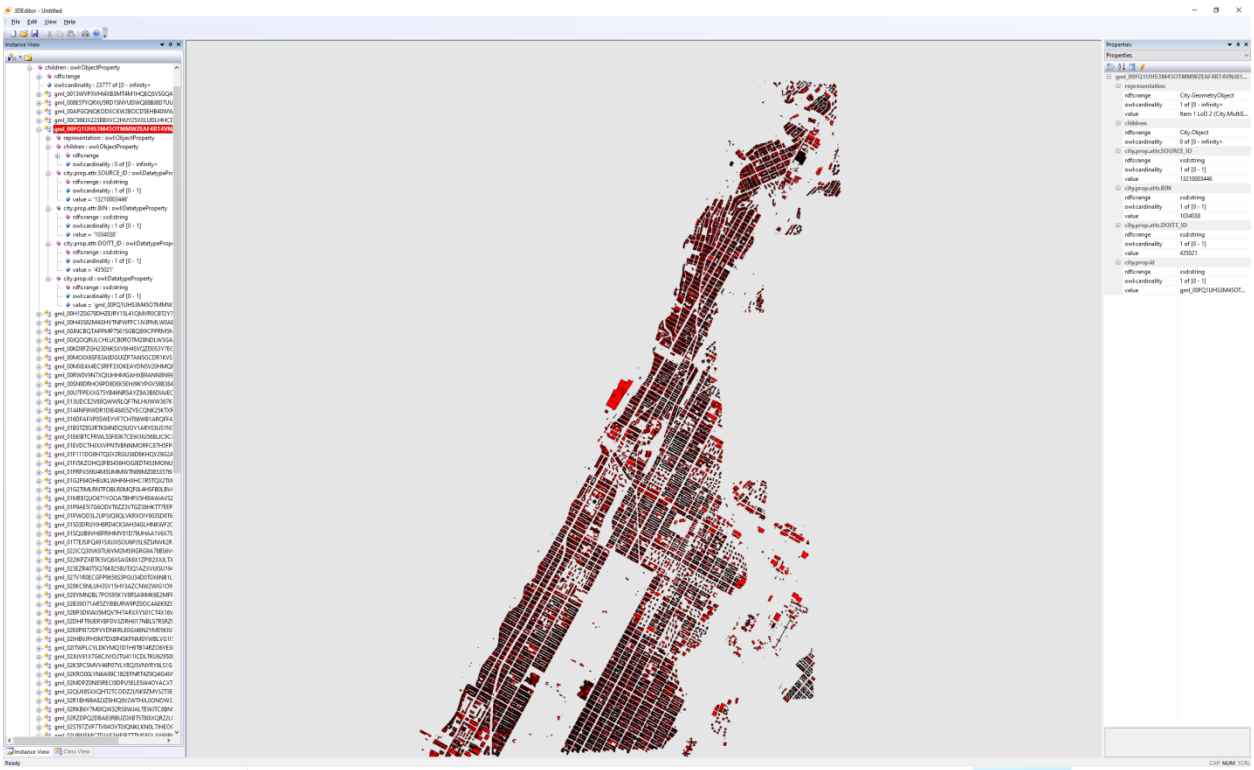


Figure 2 CityGML large example 1

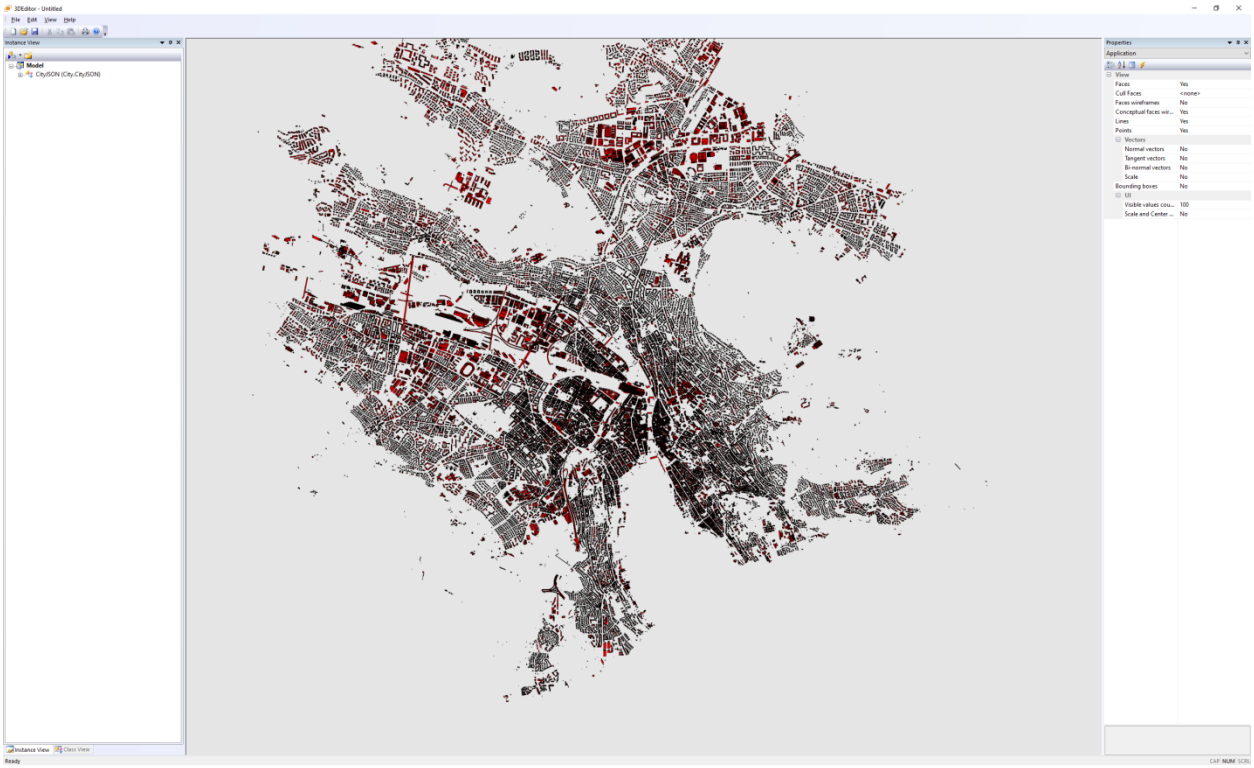


Figure 3 CityGML large example 2

During development it was recognized that much of the available content was not fully compliant with the defined schema; to enable correct testing CityGML and CityJSON models have been retrieved from public sources but specifically also from existing contacts from RDF Ltd., many of these models are protected and cannot be shared within the project but could be used for testing of the developed tooling due to existing NDA's with the software clients from RDF Ltd. Logging functionality and a test application have been developed to recognize (and where possible solve) such inconsistencies in the data. This enables a new unexpected use of the library where files can be imported and directly exported, fixing inconsistencies in the original file.

3.2 Technical Details

3.2.1 Implementation

The following core components are used and / or developed:

- C++ 11 as Library
- Parsers for XML/JSON
- Log support
- Exceptions support
- HTTP Client

3.2.2 XML / JSON schemas

The following functionalities have been developed:
Downloading schemas on demand (redirection support, e.g. obsolete/wrong Url-s)
Embedded schemas (serialization for C++ classes, data members and references (pointers))
Support for XML/JSON/UML classes.
Support for missing schemas/complex/simple data types: class:Thing/prop:string

3.2.3 Third-party libraries

The following third-party libraries (excluding commercial libraries from RDF) have been used.

- libzip
  - License: BSD
  - License · libzip

Notes: optional; can be disabled.

3.2.4 Interfaces

The following languages are supportedd for the API (excluding wrappers):

- C
- C++ (11)

3.2.5 Supported formats

The following open standard formats are supported:

- CityGML (support for Export)
- CityJSON (support for Export – in progress)
- InfraGML
- LandXML

3.2.6 Supported geometry

The following CityGML / CityJSON geometrical base classes are supported (i.e. understood):

- Solid
- Shell
- MultiSolid (no test data)
- CompositeSolid
- MultiSurface
- CompositeSurface
- TexturedSurface
- MultiGeometry
- OrientableSurface
- TriangulatedSurface
- Triangle
- TIN
- MultiCurve
- CompositeCurve
- LineString

- Polygon
- Point
- MultiPoint
- ImplicitGeometry

3.2.7 Supported materials

The following materials are supported (i.e. understood)

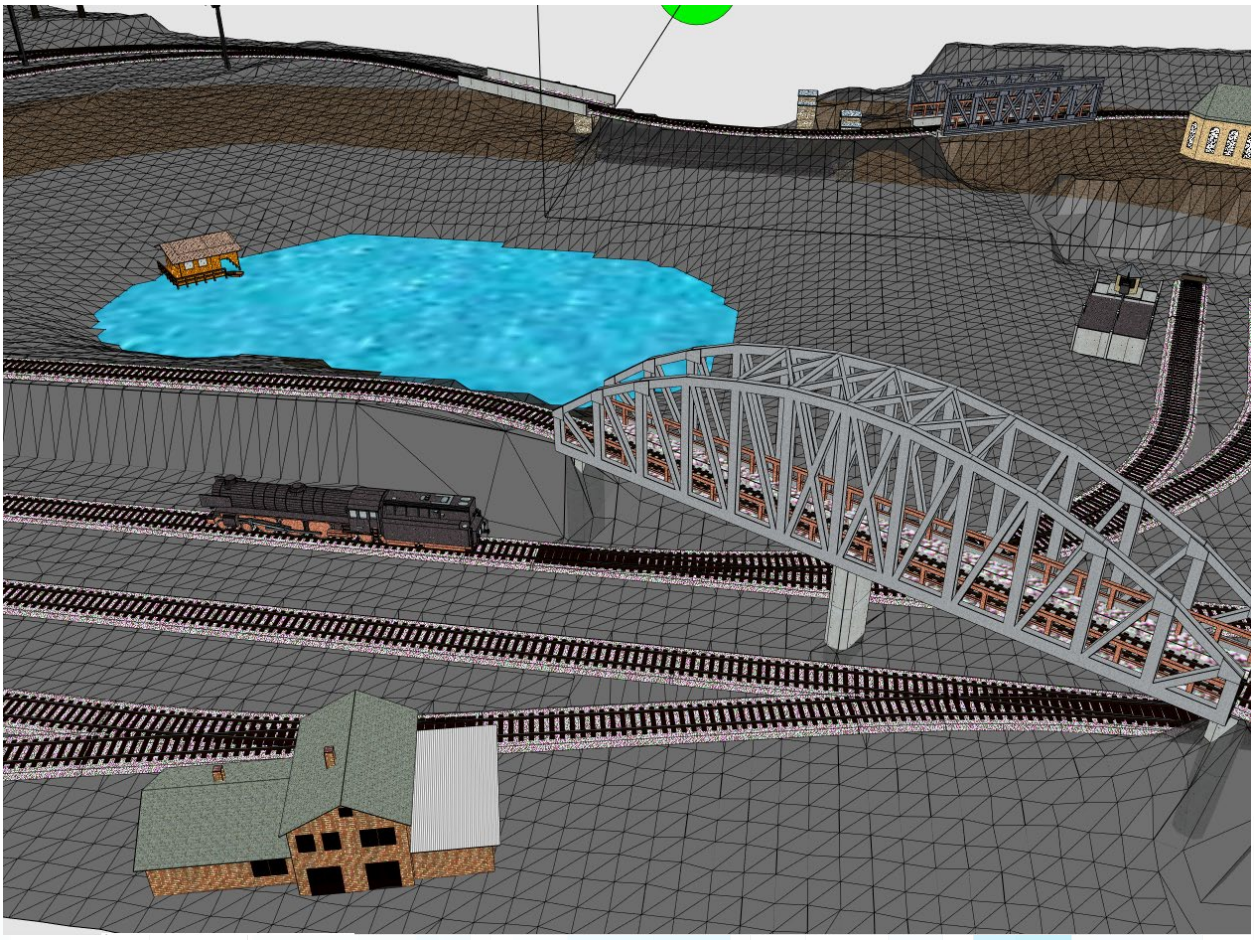
- X3DMaterial
- ParameterizedTexture
- GeoreferencedTexture (no test data)

3.3 Examples

This section shows several visiualisations produced be the new GIS library.

3.3.1 CityGML / CityJSON

CityGML/texture1/*
CityGML/texture2/*
Notes: Models have textures/colors; CityGML models can be exported.



3.1: Geo to BIM tool/procedure

Figure 4 CityJSON with textures and multi-placement

3.3.2 InfraGML

InfraGML /InfraGML_DenHaag_01.xml
Notes: there are no materials; the default color is used.

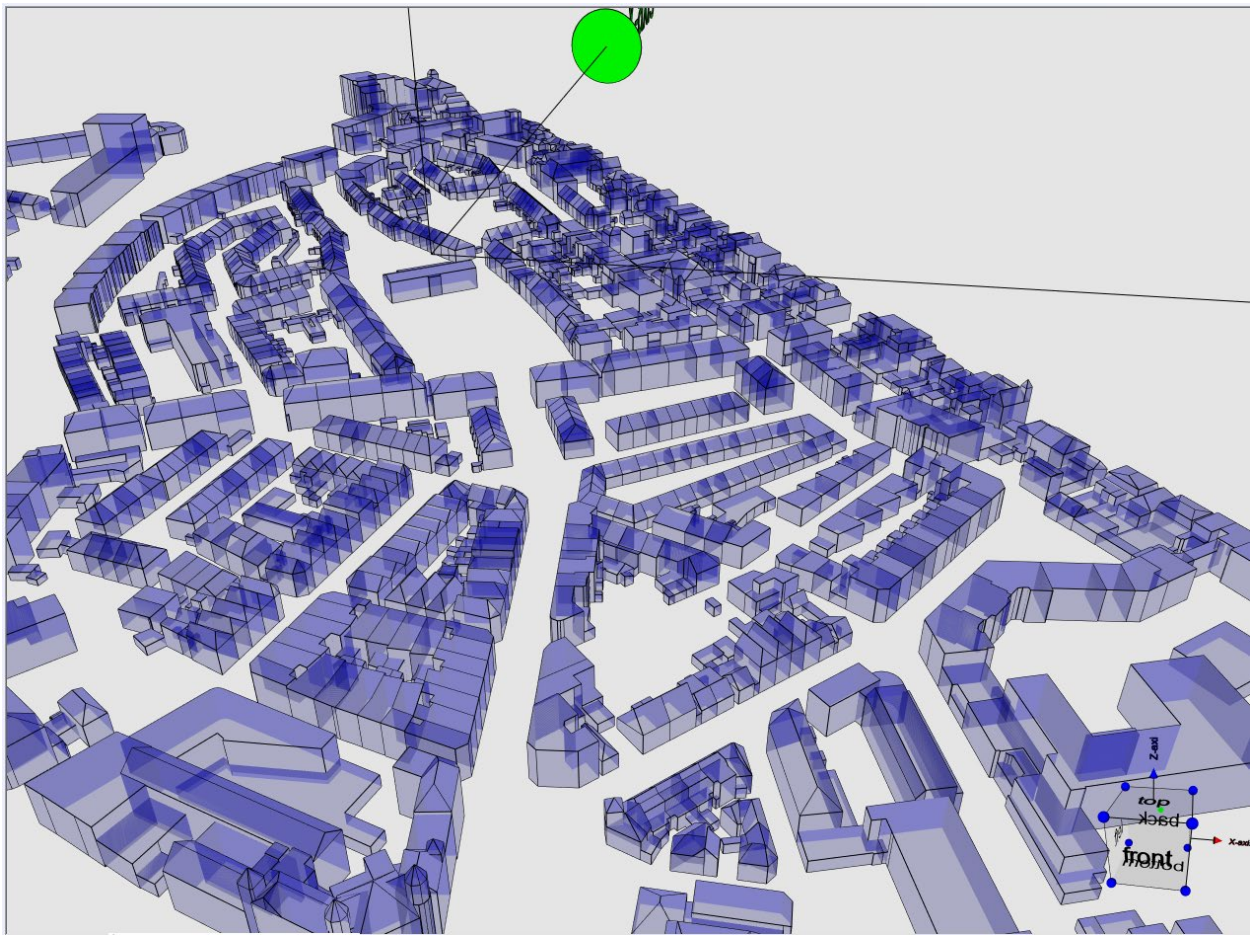


Figure 5 InfraGML example

3.3.3 LandXML

LandXML/siteops.xml

Notes: there are no materials; the default color is used.

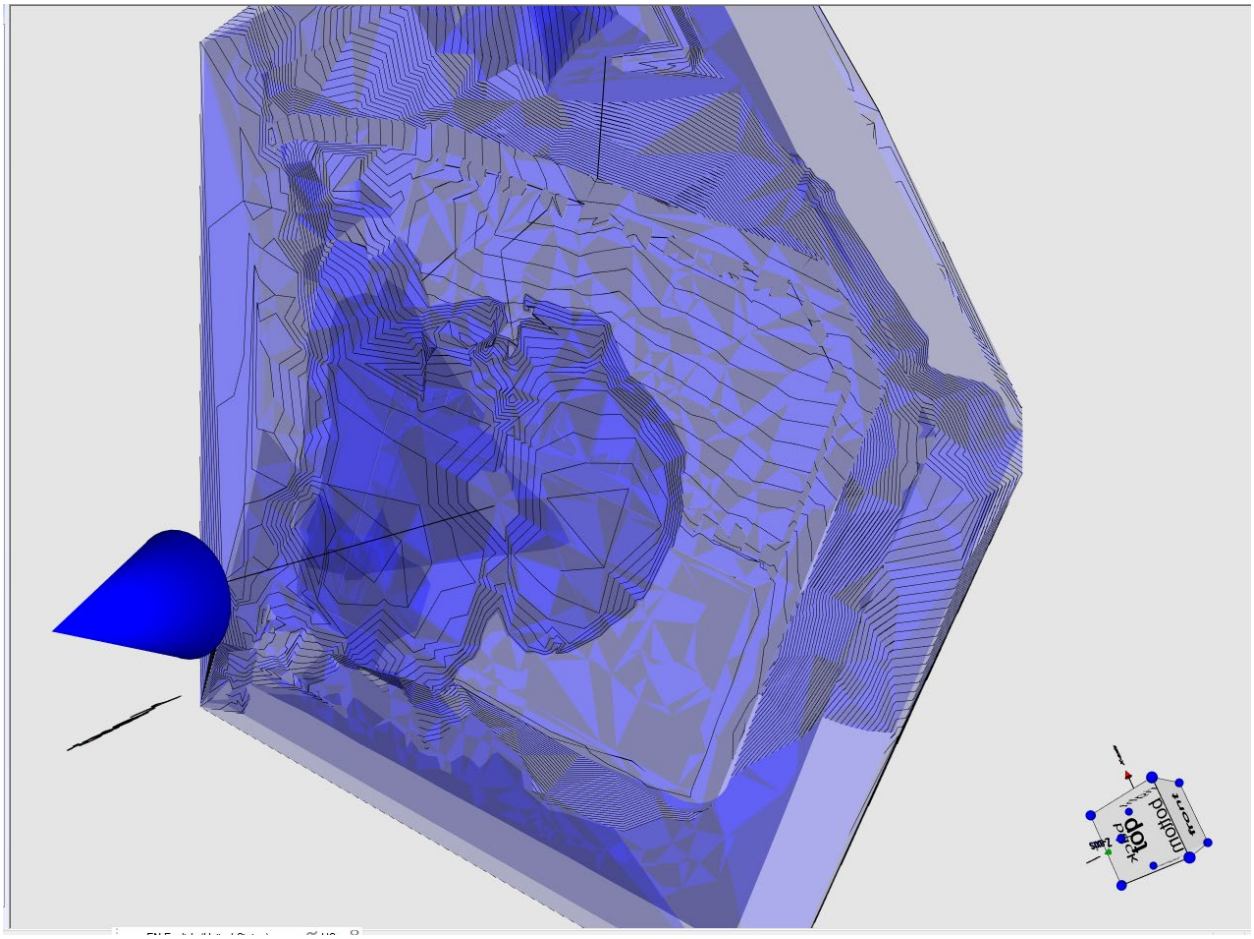


Figure 6 LandXML example

4. Geo to BIM Converter

Once both the new GIS library (GIS Engine) and existing BIM library (IFC Engine) were available, the start of the actual conversion could be developed. The converter itself is developed as an open-source solution; although the underlying libraries are not open source, the API accessing the RDF / RDFS / OWL content is a public API and the API to create the IFC file is an open ISO standard (SDAI, i.e. ISO 10303-22).

An IFC file has more structure and relations compared to CityGML (on all available LoD levels). This means that the converter has to be generating some unknown knowledge; this is true for the decomposition / containment structure, classification and also for the geometry itself. The default converter recognizes classification when and as available in the CityGML data model, i.e. walls, roofs etc.; the mapping between this classification in CityGML and the available classes (called entities) in the IFC data model (i.e. for CHEK the IFC 4 ADD2 TC1 data model) is hardcoded (see also Technical Details within this Chapter). All geometry in the original file is already converted into the GEOM ontology and a generic conversion from relevant parts of the GEOM ontology towards IFC defined entities representing the geometry is developed. Here, also the knowledge of repeating geometry as originally stored in CityGML and as such converted to the GEOM ontology is also properly converted into IFC. This allows for example a forest of trees defined in CityGML with one reference geometry to have the same minimal memory use for geometrical storage in the generated IFC file.

All non-geometrical semantics that have no IFC equivalent that can be identified as simple or complex properties are converted to dedicated IFC property sets, i.e. these are not standard IFC property sets but can be recognized as relevant semantically data by any CAD system reading the generated IFC file. For CHEK, it is very relevant that some important semantic data is stored in an unambiguous manner in the resulting IFC file. To enable this, a mapping mechanism is developed through the use of a settings text file that can be adjusted by the user. The settings file can be adjusted in Notepad or any other text editor. This mapping mechanism allows identifying simple or complex properties in the CityGML file (they could even be part of a profile or an ADE), together witht the type, property-set and property name howthey should be added in the IFC file. When the converter identifies such properties in the input file, it will generate the relevant property and property set. This could be a dedicated property set but also a property set part of the IFC definition.

4.1 Mapping - settings file

The settings text file is a TAB separated file, containing one setting per line; all values that have prefix '$' will be treated as a reserved keyword e.g. $VERSION, $MATERIAL, $PROPERTY.

The file itself is a simple text file and can be created and opened by for example Microsoft Notepad or any other simple text editor application. The text file exists of several sections, lines that start with '#' will be treated as a comment (ignored).

The used class names map back to the class names as defined within CityGML. It is possible to use a class name not defined in the core CityGML data model; this allows an ADE to define a new class name directly usable within this mapping file.

Sections:

Comment

All lines that start with '#' will be treated as a comment (ignored)

Examples:

```
### Materials ###

### Properties ###


$VERSION
```

Version/format of the file; for backward compatibility.

Example:

```
$VERSION        1.0


$MATERIAL
```

Consists of four values:

```
$MATERIAL - SCOPE - TARGET CLASS - RGBA
```

RGBA – all values are [0 - 255]; A = 0 means non-transparent material.

It could describe 'default' material for all or set of classes; it will be used when there is no material defined.

Examples:

```
$MATERIAL       $DEFAULT        $ALL    0;0;255;191

$MATERIAL       $DEFAULT        $ROOF   139;69;19;0

$MATERIAL       $DEFAULT        $WALL   128;128;128;0
```

Also, it could describe 'override' material; in which case it will be used even if there is a material defined for a given set of classes (separation is by tabs ASCII #09 and line end, i.e. ASCII #10 (+ #13)).

Examples:

```
$MATERIAL       $OVERRIDE       $ROOF   255;0;0;0

$MATERIAL       $OVERRIDE       $WALL   0;255;0;0


$PROPERTY
```

Consists of five values:

```
$PROPERTY - IFCINTEGER - PROPERTY SET - PROPERTY NAME - Human-readable PROPERTY NAME
```

It describes a property set and list of 'well-known' properties; all attributes and properties which are not described in .settings file will be added to the 'default' property set: "set_BsAttributes&Properties"

Examples:

```
$PROPERTY IFCINTEGER "set_BsSurroundingBuilding" "NumberOfFloorsAboveGround" "Number of floors above ground"

$PROPERTY IFCLENGTHMEASURE "set_BsSurroundingBuilding" "TotalHeight" "Total Height"

$PROPERTY IFCIDENTIFIER "set_BsParcel" "nationalCadastralReference" "Parcel Cadastral Reference"
```

4.2 Technical Details

4.2.1 RDF / RDFS classification

Supported RDF / RDFS classes originated from CityGML within the converter are:

- Buildings
  - Building
  - WallSurface
  - RoofSurface
  - Door
  - Window

- Features
  - VegetationObject
  - WaterObject
  - AbstractBridge
  - AbstractTunnel
  - TransportationObject
  - CityFurniture
  - ReliefComponent
  - LandUse
  - TrafficSpace
  - TrafficArea

4.2.2 CityGML-RDF / RDFS to IFC non-geometrical semantics mapping

- Buildings
  - Building => IfcBuilding
  - WallSurface => IfcWall
  - RoofSurface => IfcRoof
  - Door => IfcDoor
  - Window => IfcWindow
  - Unknown (not supported) building elements => IfcBuildingElementProxy

- Features
  - VegetationObject => IfcGeographicElement
  - WaterObject => IfcGeographicElement
  - AbstractBridge => IfcTransportElement
  - AbstractTunnel => IfcTransportElement
  - TransportationObject => IfcTransportElement
  - CityFurniture => IfcFurnishingElement
  - ReliefComponent => IfcGeographicElement
  - LandUse => IfcTransportElement
  - trafficSpace => IfcTransportElement
  - TrafficArea => IfcTransportElement
  - Unknown (not supported) building elements => IfcBuildingElementProxy
  - Implicit Representations (templates) => IfcMappedItem

4.2.3 RDF / RDFS to IFC geometrical semantics mapping

- MultiSurfaceType,
- CompositeSurfaceType,
- BoundaryRepresentation
- etc. => IfcClosedShell
  IfcProductDefinitionShape

  IfcStyledItem

  IfcCartesianTransformationOperator3D

  IfcShapeRepresentation,
  RepresentationType: PointCloud, Curve3D, Brep, MappedRepresentation, etc.

  IfcPropertySet, etc.

4.2.4 Notes & Open Issues

All Attributes / simple data type properties as well as complex data type properties are exported as an IfcPropertySet. This allows anybody using the generated IFC file to access any simple / complex data type property stored in the original CityGML / CityJSON file. The consequence is that the resulting IFC files can grow large in size; if it is found that the generated IFC files will become too large to handle part of this data could be excluded.

There is a default material (transparent blue), which is used in case there is no material or the material has texture.

One known open issue within CityJSON is that some building elements are not children of the Building and they are not exported, e.g. Chimney (LoD3_Railway.city.json.ifc).

4.3 Examples

4.3.1 Example 1

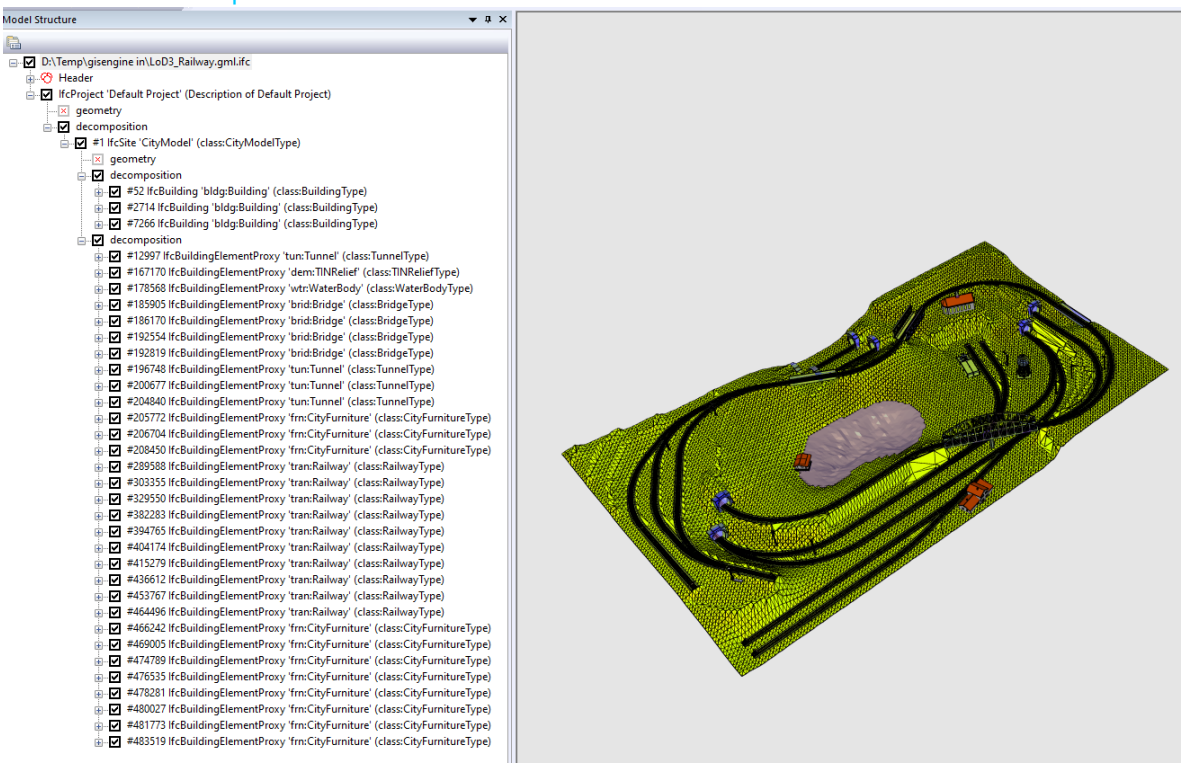


Figure 7 CityJSON converted as IFC 4 ADD2 TC1

4.3.2 Example 2

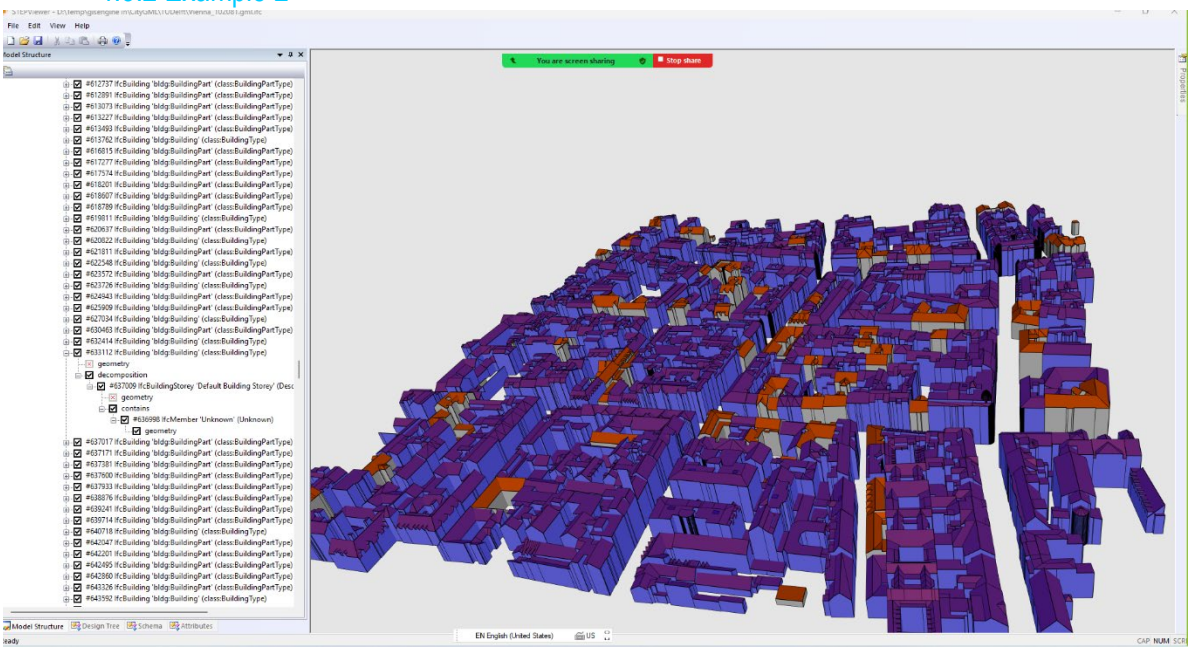


Figure 8 CityGML converted as IFC 4 ADD2 TC1

5. Software / Source Code Download Locations

The components that have been developed as part of this task are available via the following links::

- https://rdf.bg/product-list/ifc-engine/ Commercial BIM engine
- https://rdf.bg/product-list/gis-engine/ Commercial GIS engine
- https://rdf.bg/product-list/geometry-kernel/ Commercial Geometry Modelling Kernel (base for BIM and GIS engine)
- https://rdf.bg/download/3deditor-20240902.zip 3D Viewer / Editor for supported GIS open standard formats
- https://github.com/peterrdf/gml2ifc Open-source Geo to BIM converter
- https://github.com/peterrdf/gml2ifc/tree/main/x64/Release Windows Stand-Alone Application
- https://rdf.bg/CHEK/gml2ifc.html Web Assembly based Online Service
- https://github.com/peterrdf/stepengineexamples/tree/main/bin/32bit/WASM distributable package

6. Conclusion

The use of the underlying existing commercial grade libraries Geometry Modeling Kernel and IFC Engine allowed the development of a Geo to BIM converter with high TRL (Technology Readiness Level).

The generic setup of the GIS engine allowed developing support for all versions of CityGML and CityJSON in serializations. Although originally not expected, there was also the need to support GML files as well as support for CityGML and CityJSON files that are invalid according to the specification.

The resulting IFC files are validated with tooling created in the context of WP2, through the validation service of bSI and checked by reading them into CAD applications like Revit. The results of these validations look promising regarding the quality of the output.

7. References

7.1 List of Figures

7.2 List of used abbreviations

BIM	-	Building Information Model
bSI	-	buildingSMART International
DBP	-	Digital Building Permit
DoA	-	Description of the Action
IFC	-	Industry Foundation Classes
GA	-	Grant Agreement
GEOM	-	RDF / RDFS Ontology for Representing Geometry
GIS	-	Geographic Information System
GML	-	Geographic Markup Language
RDF	-	Resource Description Framework
RDFS	-	Resource Description Framework Schema
WP	-	Work Package