AUTOMATIC CONVERSION OF CITYGML TO IFC

N. Salheb[*], K. Arroyo Ohori, J. Stoter

3D Geoinformation, Department of Urbanism, Faculty of the Built Environment, Delft University of Technology,
Julianalaan 134 Delft 2628BL the Netherlands

KEY WORDS: CityGML, IFC, Conversion, GeoBIM

ABSTRACT:

The trend of increased usage of both BIM and 3D GIS and the similarity between the two has led to an increase in the overlap between them. A key application of such overlap is providing geospatial context data for BIM models through importing 3D GIS-data to BIM software to help in different design-related issues. However, this is currently difficult because of the lack of support in BIM software for the formats and data models of 3D Geo-information. This paper deals with this issue by developing and implementing a methodology to convert the common open 3D city model data model into the most common open BIM data format, namely CityGML (Gröger et al., 2012) to IFC (buildingsmart, 2019b). For the aim of this study, the two standards are divided into 5 comparable subparts: Semantics, Geometry, Geographical coordinates, Topology, and Encoding. The characteristics of each of these subparts are studied and a conversion method is proposed for each of them from the former standard to the latter. This is done by performing a semantic and geometrical mapping between the two standards, converting the georeferencing from global to local, converting the encoding that the two standards use from XML to STEP, and deciding which topological relations are to be retained. A prototype implementation has been created using Python to combine the above tasks. The work presented in this paper can provide a foundation for future work in converting CityGML to IFC. It provides an insight into the relationship between the two standards and a methodology for the conversion from one to the other, and the process of developing software to perform such conversion. This is done in a way that can be extended for future specific needs.

1. INTRODUCTION

Since 2008 there was an increase in the use of Building Information Modeling (BIM) within the construction industry. Similarly, Geographic information systems (GIS) have been increasingly used to generate detailed 3D data, and in particular 3D city models. Both GIS and BIM can, therefore, provide 3D data, but they differ in terms of their characteristics and focus. As a result, BIM models are more detailed and semantically rich than GIS. On the other hand, GIS has less detailed but more updated datasets describing the environment in a wider area (Arroyo Ohori et al., 2018). The trend of increased usage of both BIM and 3D GIS and the similarity between the two has also led to an increase in the overlap between them.

A possible application of such overlap is providing geospatial context data for BIM models through importing 3D GIS to BIM software (Figure 1). Currently, this is difficult because of the lack of support in BIM software for the formats and data models of 3D Geo-information. This paper aims to address this issue by providing a methodology to convert the common open 3D city model data model into the most common open BIM data format, namely CityGML (Gröger et al., 2012) to IFC (buildingsmart, 2019b).

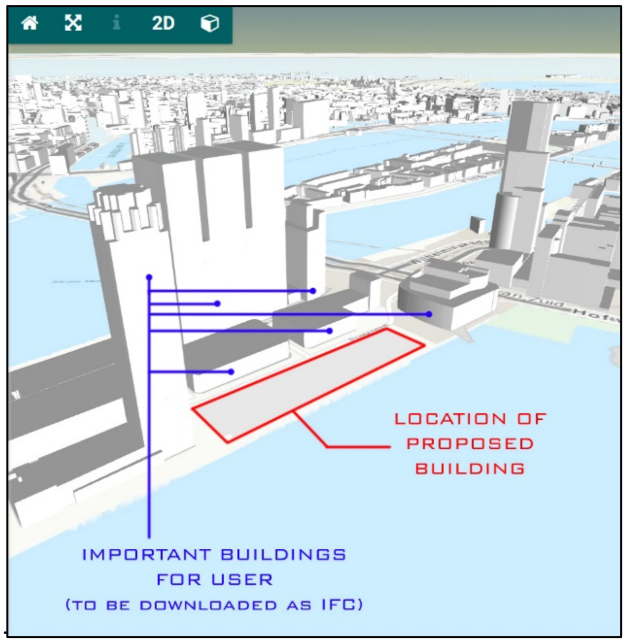
2D
LOCATION OF PROPOSED BUILDING
IMPORTANT BUILDINGS FOR USER
(TO BE DOWNLOADED AS IFC)

Figure 1. Contextual design of a new building requires the BIM models of the immediate surroundings

[*] Corresponding author

2. BACKGROUND

Both CityGML and IFC provide a representation of different aspects of 3D models, including semantics, geometry, topology, and appearance. However, they differ widely in the way they store and represent this data. Hereafter, the characteristics of each standard are analyzed, divided into 5 main components in order to make it clear how to compare and later convert between each of them. These components are: **Encoding, Semantics, Geometry, Coordinates, and Topology.**

2.1. CityGML

Encoding: CityGML is an application schema for GML 3 that is based on XML. It provides a common definition of basic entities, attributes, and relations of a 3D city model. It has a tree representation of data that will create a hierarchal structure that reaches down to individual features and attributes. In CityGML, objects can be represented in five different levels of detail, where objects become more detailed with the increased LOD and it differs regarding its geometry and thematic representation.

Semantics: In CityGML, features are an abstraction of real-world objects and, semantically, it is modeled by classes that are specified using UML notation. These geographic features may have an arbitrary number of spatial and non-spatial attributes. (Kolbe - 2009 - Representing and Exchanging 3D City Models with Ci.Pdf, n.d.**).**

Geometries: The geometries of geographic features are represented as objects that have an identity and further geometric substructures. Buildings and building objects' geometrical representations are defined implicitly. This means that an object is defined by attributes that define its sub-elements, which are then combined to form the complete object.

Coordinates: In CityGML all coordinates belong to a world coordinate reference system (CRS) and local transformations are not allowed, which means that geometry belongs to exactly one fixed place in space.

Topology: The topology model of GML3 follows a well-defined relational schema between elements, which is a line of full decomposition of n-dimensional topological primitives into (n-1)-dimensional primitives, which again are decomposed down to the level of nodes (0D) (Kolbe, 2009).

2.2. IFC

Encoding: IFC is an open standard and format to exchange BIM data, it provides a very detailed semantic model for 3D building (Eastman et al., 2011; Shen et al., 2010). The IFC architecture has an entity-relationship model that is based on Express relations. It consists of hundreds of entities that have a hierarchy that is based on object inheritance relation (buildingsmart, 2019b). The most common file format is IFC-SPF which has a STEP encoding which is a plain text format that is human readable and more compact compared to XML.

Semantics: Since the scope of IFC is restricted to buildings and sites, no topographic feature classes like terrain, vegetation, water bodies, etc. are included. IFC is a semantic model like CityGML, but with a different scope and at a different scale (El-Mekawy et al., 2012, p. 160). And unlike CityGML there is no formal approach adapted for multi-scale representation (Borrmann et al., 2013, p. 1).

Geometries: There are distinctive geometrical models characterized in IFC for example: CSG (Constructive Solid Geometry), BRep, or Sweeping. The semantic implementations of objects are unambiguously mapped in IFC with a strict separation between geometry and semantics. In IFC 2x3 there are 653 entities, however, most of these classes are used to create a spatial relation between elements and their geometric representation.

Coordinates: IFC has classes that can describe the information required for georeferencing. IFCSite (buildingsmart, 2019a) can have the information of a geographic reference point for the project site in WGS84 with Longitude, Latitude, and Elevation. If these values are given, it provides absolute placement in relation to the real world. The geographic reference point would be the location of the point (0.,0.,0.) of the local placement of the IFCSite.

Topology: The IFC structure is designed to support dynamic modeling, that provides the users with the flexibility to represent their building data. All the objects in IFC can be created using some core elements, these core elements contain the general information. This flexibility in IFC results in different possible ways of connecting two different elements in IFC.

3. METHODOLOGY

Figure 2 shows an overview of the methodology that we have developed to convert CityGML to IFC. The first step in the methodology is to study and provide a theoretical conversion for each of the different components of both standards, these are: semantics, geometry, topology, encoding, and georeferencing. Next, all these conversion requirements are combined in one implementation and then implemented via an incremental development process starting from converting a simple dataset to a complete dataset and different datasets. Next, a collection of software is selected to check the results. By checking the results in this software, the methodology is improved accordingly, and the implementation is debugged in an iterative process. At the end of this process, the conversion implementation is finished.

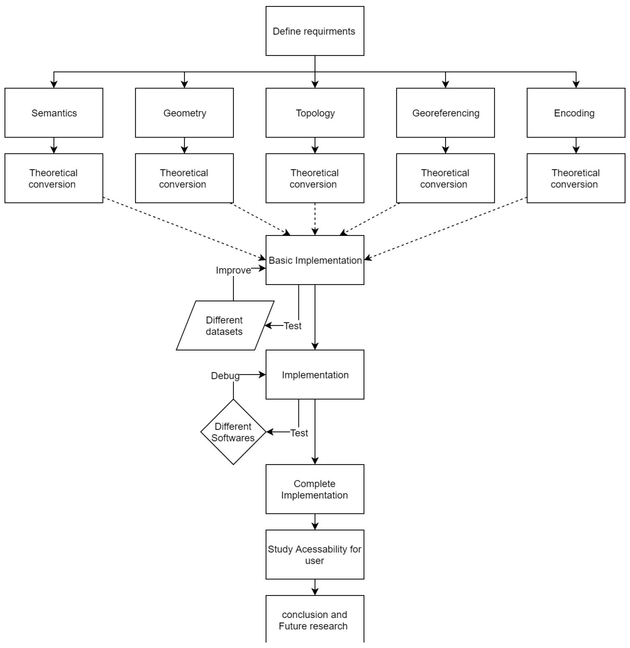

Figure 2. Overview of methodology

3.1. Encoding

Encoding is the first major obstacle that can obstruct the conversion. Here, a software is developed using Python to read the source XML data, parse the data, apply the transformation, and write the STEP file (STEP, 2017) for IFC. The development of the Python tool is done incrementally, for example: starting from converting one single CityGML element (such as a wall) to IFC, then converting a complete building and then including other buildings and other city objects. The difference between the two data models requires remodeling of the source elements from the hierarchical encoding of CityGML to their counterparts of the non-hierarchal encodings in the IFC data model.

3.2. Geometry

The CityGML geometry of buildings consists mainly of 2D surfaces. On the other hand, IFC objects are built using different geometry representations including: sweep volumes, explicit faceted surface models, and CSG (Arroyo Ohori et al., 2017). To be able to convert an element from CityGML to IFC; the accurate matching geometry should be created. Since the focus of this research is on converting buildings classes, the representations of IFCWall and IFCSlab are particularly interesting since it is possible to model the whole LOD2 building geometry with these two classes. In IFC 2x3, the use of 'SweptSolid' and 'Clipping' representations is supported for these classes. Also, the general representation types 'Brep', 'SurfaceModel', and 'BoundingBox' are allowed (buildingsmart, 2017). In this research, the direct conversion from 2D surface in CityGML to 'SurfaceModel' in IFC is performed.

Figure 3 shows the conversion process, where the same process is repeated for every surface element in CityGML. Here, the source file in CityGML is a ground surface that is defined by a linear ring that consists of 6 points:

```
      <bldg:boundedBy>
        <bldg:GroundSurface>
          <bldg:lod2MultiSurface>
            <gml:MultiSurface>
              <gml:surfaceMember>
                <gml:Polygon gml:id="RCID_98b8908e-480d-4e67-8cb8-
2967c0d78523">
                  <gml:exterior>
                    <gml:LinearRing gml:id="RCID_98b8908e-480d-4e67-
8cb8-2967c0d78523_E_1_1">
                      <gml:posList>94713.250000000000000000
433753.299999999990000000 -1.226650000002190000
94712.259999999995000000 433751.789999999980000000 -
1.226650000002190000 94700.619999999995000000
433759.409999999970000000 -1.226650000002190000
94703.250000000000000000 433763.419999999980000000 -
1.226650000002190000 94709.589999999997000000
433759.250000000000000000 -1.226650000002190000
94707.960000000006000000 433756.770000000020000000 -
1.226650000002190000 94713.250000000000000000
433753.299999999990000000 -1.226650000002190000</gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:MultiSurface>
          </bldg:lod2MultiSurface>
        </bldg:GroundSurface>
```

Figure 3. Example of CityGML GroundSurface geometry to be converted to IFC

The total number of values in gml:posList is 21 defining the coordinates of 7 points because the end of the ring is the same as the beginning. Here, using element tree tree.findall('.//{%s}posList' % ns_gml) is used to a create list of IFCCartesianPoint entities, where IFCCartesianPoint is a point defined by a three-dimensional Cartesian coordinates system. The list of these points forms an IFCFaceOuterBound, which defines the outer boundary of the face, i.e. IFCFace. To define the geometric representation of the face the following entities inheritance is defined:

IFCOPENSHELL
↳ IFCSHELLBASEDSURFACEMODEL
↳ IFCPRODUCTDEFINITIONSHAPE
↳ IFCSLAB

The resulting geometry is compliant with the requirement of the IFC standard and provides an accurate syntactical representation of the external shell of the building. But it is lacking the real volume of the emanates (i.e. thickness) as would probably be found in IFC datasets that are generated from BIM models.

3.3. Coordinates

To georeference the resulting IFC data file out of the source CityGML files, the following process is followed: Firstly, a reference point for the model is created, this is done by iterating over all the points in the model and selecting the minimum value for all the points. For example, in Figure 4, a reference point for a GroundSurface in CityGML is created, with an EPSG:28992 coordinate system.

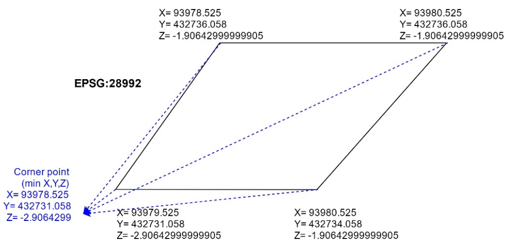

Figure 4. Creating a reference point based on minimum values

Secondly, the values of all points are converted to local referencing in relation to the reference point. The local referencing is calculated by subtracting the values of all points from the reference point value, which is shown in Figure 5.

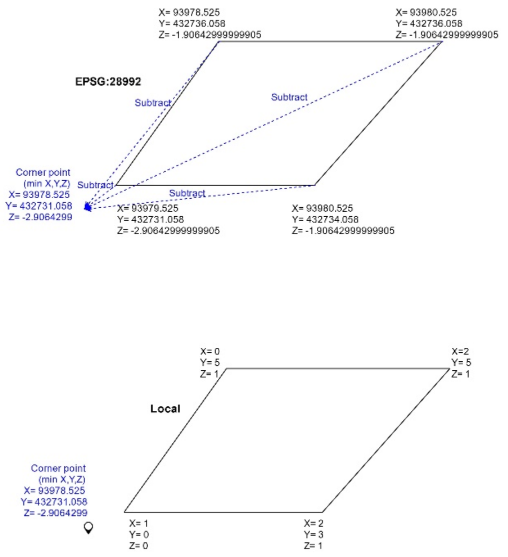

Figure 5. Converting calculating local referencing to all points

Lastly, latitude, longitude in WGS 84, and elevation of a project are dedicated to the IFCSite class (Diakité, 2018). BIM model coordinates are usually stored in an IFC transformation file. Since the EPSG:28992 system in the Netherlands is based on XY coordinates that are aligned with longitude and latitude, the rotation degree is not needed (Diakité, 2018).

3.4. Semantics

For semantics mapping, different criteria are used to create semantics mapping between the IFC domain and the CityGML domain, including; checking if there is an existing class in IFC that matches the source CityGML class, and the possibility to match the geometry, and the semantic matching practiced in different research. For example, in Kavisha (2015), the classes and notations of the two data models are collected and semantic mapping is created. In addition, the different practices in software that deals with this problem, most notably in FZK Viewer (Hütter, 2016). That helped to match the semantic terminologies of objects and classes used in the data models, which are shown in Table 1:

CityGML	**IFC**
AbstractBuilding	IfcBuilding
-GroundSurface -FloorSurface -CeilingSurface	IfcSlab -GroundSlab -FloorSlab -CeilingSlab
RoofSurface	IfcRoof
-WallSurface -InteriorWallSurface	IFCWall -Interior Wall -Exterior Wall
WallSurface	IfcCurtainWall
GenereicCityObject	IfcBuildingElementProxy
SolitaryVegetationObject	IfcBuildingElementProxy
Opening Door Window	IfcOpeningElement IfcDoor IfcWindow
BuildingInstallation	IfcBeam, IfcColumn, IfcCovering, IfcStair, IfcRailing, IfcRamp

Table 1. IFC-CityGML Mapping (Kumar & Saran, 2015), own work

Based on this table, CityGML models in different LODs can be converted to IFC while preserving most of their semantic information (El-Mekawy et al., 2012). Based on the above proposed semantic mapping, the semantic transformation is performed based on the mapping in Figure 6:

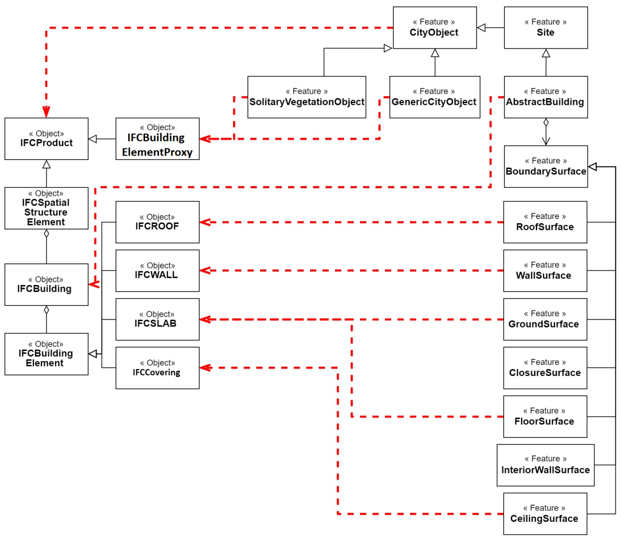
« Feature » CityObject
« Feature » Site
« Feature » SolitaryVegetationObject
« Feature » GenericCityObject
« Feature » AbstractBuilding
« Object» IFCProduct
« Object» IFCBuilding ElementProxy
« Feature » BoundarySurface
« Object» IFCSpatial Structure Element
« Object» IFCROOF
« Feature » RoofSurface
« Object» IFCBuilding
« Object» IFCWALL
« Feature » WallSurface
« Object» IFCSLAB
« Feature » GroundSurface
« Object» IFCBuilding Element
« Object» IFCCovering
« Feature » ClosureSurface
« Feature » FloorSurface
« Feature » InteriorWallSurface
« Feature » CeilingSurface

Figure 6. Semantic mapping application

3.5. Topology

It is important to show the spatial structure of the project elements using ‘IfcRelAggregates’. This is particularly important for some BIM software to read the IFC file correctly. To complete this relation IfcRelAggregates is used to represent the physical containment of the buildings in the IFCProject, this creates a certain level for the building in the spatial structure, which allows the use of ‘IfcRelContainedInSpatialStructure’ to assign sub-elements of the building. It is worth noting that an element can be assigned once to a certain level of spatial structure. However, using IfcRelContainedInSpatialStructure spatial containments can be assigned on multiple levels. Hence a wall, for example, can be contained in both a building and a building story. For the purpose of this study no additional spatial relations are created beyond the relations that originally exist in the CityGML dataset (aggregation and containment), hence no association of the elements to a building story for example.

The complete methodology resulting model is shown in Figure 7:

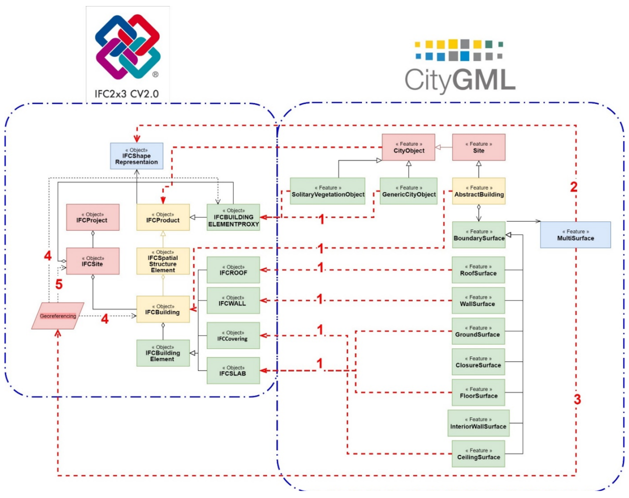
IFC2x3 CV2.0
CityGML
« Object» IFCShape Representaion
« Feature » CityObject
« Feature » Site
« Feature » SolitaryVegetationObject
« Feature » GenericCityObject
« Feature » AbstractBuilding
2
« Object» IFCProject
« Object» IFCProduct
« Object» IFCBUILDING ELEMENTPROXY
1
« Feature » BoundarySurface
« Feature » MultiSurface
1
4
« Object» IFCSite
« Object» IFCSpatial Structure Element
« Object» IFCROOF
1
« Feature » RoofSurface
5
« Object» IFCWALL
1
« Feature » WallSurface
Georeferencing
4
« Object» IFCBuilding
« Object» IFCCovering
1
« Feature » GroundSurface
« Object» IFCBuilding Element
« Feature » ClosureSurface
« Object» IFCSLAB
1
« Feature » FloorSurface
3
« Feature » InteriorWallSurface
« Feature » CeilingSurface

Figure 7. The complete methodology resulting data model

In the figure above the following transformations are shown numbered:

1. Semantic mapping from CityGML features to IFC objects.
2. Creating Geometry resources for IFC objects based on source CityGML geometry.
3. Creating Georeferencing point from the CityGML dataset.
4. Georeferencing IFC objects.
5. Storing Georeferencing information in the IFCSite object.

4. IMPLEMENTATION

The main implementation part consists of a program named "CityGML2IFC.py" it is a script file written in Python 3. When compiled, the program will convert a source file in CityGML to the destination file in IFC. The complete code and license information is be found on GitHub here: https://github.com/nsalheb/CityGML2IFC

Table 2 shows the modules that are imported and, which should be preinstalled before running the program:

Module names	**usage**
XML.etree.ElementTree	Is used here for parsing the XML data
os	To interact with the operating system where the computer is running, for example, reading time and file bath.
time	To read the current time and stored in the created IFC files
itertools	Is used to create a hashtagged unique id with an incremental value starting from a given value
sys	Is used to allow files to be written on the hard disk
numpy	To perform mathematical operations such as finding minimum value or subtract arrays
UUID	To automatically generate unique IDs
pyproj	To convert the projection of the resulting file

Table 2. Necessary modules for the Python program to run

The function 'CityGML2IFC (path,dst)' is the main function of this application, in which all the other functions are called. Here, path refers to the source CityGML file and dst is the destination IFC file. The main operations that are performed within CityGML2IFC(path,dst) are described as follows. The source CityGML file is parsed using 'xml.etree.ElementTree'. Next, the CityGML version is identified based on the root tag. The namespaces are created based on the CityGML version as dictionaries with keys and value. Next unique ids are generated for every object using the function 'guid()'which will automatically generate a unique id for every time it is called. For example: 'dcdc161f86a246cfb37dcb'. The results from the conversion are printed in a destination file. With the format: 'dst.ifc'. The file dst.ifc starts with the mandatory header part with the following information: file description, name, and schema. In which other IFC compatibility formatting is added such as time and file destination. Followed by the data part. (STEP-File, ISO 10303-21, 2017). Which consists of a sequence of entities, where each line represents a different entity. The names of these entities are defined by a sequence of numbers that is generated using a counter starting from 1000.

Other tools the tools that are used in this research are:

1. FME: to view and select and apply basic transformations on CityGML and IFC datasets.
2. FZK viewer: useful and fast to view data for both formats; CityGML and IFC. Moreover, it can apply basic transformations between the two data formats which are useful for testing the developed transformation.
3. ArchiCAD and Revit: both are BIM software and they are the most commonly used by users in the BIM world. This software is used to visualize and test the resulting IFC datasets. Moreover, they are used to produce samples for IFC data.
4. IfcCheckingTool: which is an analysis tool for checking the semantic and syntactic correctness of IFC data.
5. A set of BIM software to test the results on, namely: FZK Viewer, DDS-CAD viewer, Arreddo BIM viewer, BIM Vision, ArchiCAD, Revit.

5. VALIDATION

5.1. Rotterdam 3D 2.0

To test the implementation, different CityGML datasets rea used, particularly Rotterdam3D 2.0, This data is in LOD2 and based on both BAG and Rotterdam-Height model (*Hoogtebestand)* it contains the following features: Buildings, Terrain model, Trees, Street lanterns, Underground Cables, Other specially created city objects such as the Erasmus bridge.

5.2. Filtering the Dataset

IFC models are generally more rich and detailed datasets than CityGML with regards to buildings. However, IFC2x3 does not support some feature data types such as; Vegetation and Textures. Therefore, these features are ignored within the scope of the project. For these features that require to be filtered, the filtration process is done automatically because of the implementation method of choice, this is done as follows; The software will look for features with a certain tag (say: building) and convert these features. If a future tag is not incorporated in the program, then it will be automatically left out of the conversion (say: vegetation). Another filtration of the data is done based on areas, feature type, or featured ID.

5.3. Incremental development

Incremental development was done to make the methodology complete. After every test, the methodology was adjusted, and the conversion was developed to reach the complete software hence the "incremental "description.
This procedure to check the results is as follows:

1. FZK viewer: check if the results shoes probably on it FZK viewer (visual inspection).
2. Check with BIM viewer software (DDS-CAD viewer, Arreddo BIM viewer).
3. Check with Building information modeling software (Revit, ArchiCAD).
4. FZK viewer: check message log, if any message report
5. Use IfcCheckingTool tool to see advanced errors
6. Check with users.

In the beginning, initial conversion for LOD2 buildings was performed, the resulting IFC building was able to be viewed on FZK viewer after ignoring the error messages. However, for one building there were 32 errors found divided into 7 categories. These errors were a result of inaccurate conversion and indicated certain areas of the program to be improved by resolving these errors. Table 3 describes these errors and how they were solved.

Error type	**Error name**	**How the error is solved**
Data prepare	Illegal GUID found, 10 errors:(in every element including 2 times for both IFCsite and IFCbuilding)	Some elements had incorrect GUID, for example, because of too many characters in GUID or including a space in the name. This problem was solved by giving accurate IDs for all elements. As shown in the example below: 'IFC-wall'→ '36d1601925d54a5ca6a4cd'
EccoError	Runtime error: incomplete assignment, 7 errors	In STEP encoding; when assigning an object to multiple other objects the comma should be removed at the end of the list as shown in the example below: 3. [#109,#110,] → [#109,#110]
EccoError	Runtime error: unresolved reference, 2 errors	References to unexacting objects were removed
EccoError	Runtime error: missing parameter for construction type IFCRoot, 1 error	For unknown parameters, the sign: "$" is added. To avoid these kinds of errors: missing parameter
EccoError	Runtime error: too many parameters	The number of parameters should be accurate according to the IFC standard.
EccoError	Runtime error: parameter expected	For unknown parameters, the sign: "$" is added. To avoid these kinds of errors: parameter expected
MapView_002	(can't find file material in FZK program)	This is a program-related error, When the capabilities of a program, in this case, FZK viewer, are unable to view a defined material in the data file.

Table 3. Errors after initial conversion and how they were solved

After fixing the above errors, FZK viewer was able to view the results, unlike other software. Also, noticeably some errors are related to the program itself, for example, a missing material is considered an error in FZK while in other viewers is not. This error is fixed by adding material entities to the files and creating a relation between the different entities.

The IfcCheckingTool is an analysis tool for checking the semantic and syntactic correctness of IFC data. The check considers IFC Schema versions as of IFC2X3 in the file formats SPF (STEP Physical File) and ifcXML. In an automatically generated interactive report, the results can be sorted according to different criteria. As far as possible, a hyperlink to the corresponding definition in the IFC specification is output for each error, and the error within the instance document can be displayed via an EXPRESS navigation window.

After running the tool on the above city object that does not contain any errors according to FZK message log, then an error report is generated. According to the report, there are 68 errors, these errors consist mainly of a missing definition for example:

- No view definition in the header
- No material definition

The later problem was fixed by assigning different materials for every kind of element: (wall, roof, ground). However, adding materials increased the size of the file exponentially and can be ignored if not needed. Adding correct IFCRELAGGREGATES has fixed the problem with the conversion making the files readable with all the tested software. With ArchiCAD and Revit you can add elements to the resulting file. With Revit, you can even edit the resulting file. Basic editing with Revit includes moving elements such as walls and roofs, copy and paste elements, applying an array copy, and creating facades and sections. Revit is also able to provide a list of found errors. These errors were also traced and fixed (as much as possible in the final conversion).

6. CONCLUSIONS

In this paper we have described the methodology that we have developed to convert CityGML to IFC. The methodology is a result of trial and error in which a validation process of intermediate result splayed a crucial role.
Form our research, we can conclude that taking controlled CityGML data set as input (Rotterdam 3D for example), the methodology can convert the data set to a semantically accurate IFC2x3 data set that is readable by all BIM software and editable with some (e.g. Revit). The ability of a BIM modeling software also depends on the ability of the software itself to deal with IFC files. Besides, the resulting BIM models are geographically

referenced with a local coordinate system. And it has the necessary topological relations to be accurate schematically.

Some attributes are carried to the IFC file from CityGML. However, attributes are dataset dependent and the program should be edited accordingly for each different dataset. IFC data model is expected to provide enough ability to retain most of the attributes from CityGML in a straightforward way, but that requires checking which attributes are to be retained. The resulting semantic conversion is shown in Figures 8 and 9.

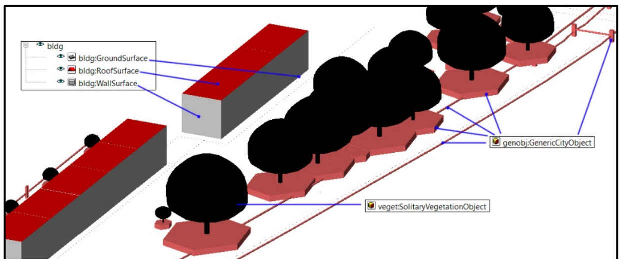

Figure 8. Source CityGML example dataset

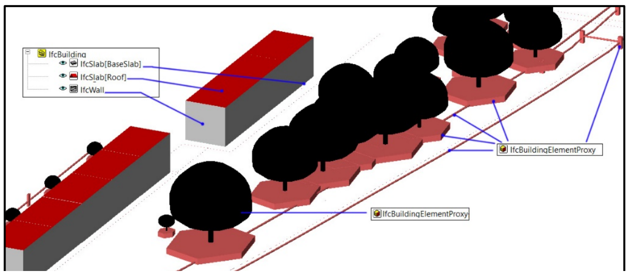

Figure 9. Resulting IFC dataset

This method to convert CityGML to IFC has different applications. The resulting IFC models could give context to designers directly in their software. This would help them in multiple design-related issues such as visualizing the shadow that their new building casts on neighboring buildings, assessing quickly whether the gardens of neighbors are visible, and detecting any clashes between a new project and its surroundings. This conversion can also encourage the production of BIM models that are possible to be prepared and later be exported to CityGML. Converting CityGML to IFC can also be helpful for creating complete BIM models for buildings that require one. Another application is the use of the resulting models to create a simplified BIM model of buildings and then develop the models using BIM software to create a thematic representation of buildings.

6.1. Discussion

This research provides a basic framework of conversion from CityGML to IFC. It is possible with additional work and adjustment to extend the work to include more feature classes and other versions of CityGML and IFC.

It is clear from the research and practice that the complexity of IFC also comes with flexibility, in contrast to the strict rules of CityGML. This leads to the conclusion that there could be different ways in terms of semantics and geometry to convert some elements from CityGML to IFC, which can lead to a different result.

Different BIM software deals with IFC data in different ways. This was helpful to provide different readings to debug the errors. It was evident also that commercial software such as Revit expected only completely accurate IFC models to be imported, unlike free software such as FZK Viewer. This could be because commercial software tries to push its own proprietary data formats.

6.2. Future work

It is important to make the conversion adequate to the needs of the user so that unnecessary information is left out from the conversion to reduce file size and for simplicity. One approach to be taken is to make the conversion as complete as possible by including all the possible information. Next, we can leave out parts of this complete conversion according to the user needs, this can be done by developing specialized tools or provide the users with the possibility to run the software with different options; "with material" and "without material" where the later create a smaller in size files.

This study presents a methodology of conversion with certain assumptions of the state of the source CityGML data. Any changes with the source data require some adjustments on the methodology. One important limitation that could be faced with other data sets is the geographical reference system. This research is done on data with an EPSG:28992 coordinate system. Data that belongs to a different coordinates system would require some adjustments accordingly. The study focuses on producing data in IFC2x3 format, while IFC4 is available since January 2019 and it is expected to become more prevailing in the upcoming period hence the required adjustment should be considered.

ACKNOWLEDGEMENTS

The authors would like to thank Ordnance Survey GB (https://www.ordnancesurvey.co.uk) and 1Spatial (https://1spatial.com/) for sponsoring the publication of this paper.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 Research & Innovation Programme (grant agreement no. 677312, *Urban modelling in higher dimensions*).

REFERENCES

Borrmann, A., Kolbe, T. H., Donaubauer, A., Steuer, H., & Jubierre, J. R., 2013: TRANSFERRING MULTI-SCALE APPROACHES FROM 3D CITY MODELING TO IFC-BASED TUNNEL MODELING. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, II-2/W1, 75–85. https://doi.org/10.5194/isprsannals-II-2-W1-75-2013

buildingsmart., 2019a: IFCSite. http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcproductextension/lexical/ifcsite.htm

buildingsmart., 2019b: Start Page of IFC2x3 Final Documentation. http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm

Diakité, A., 2018: About the Geo-referencing of BIM models. 13.

El-Mekawy, M., Östman, A., & Hijazi, I., 2012: An Evaluation of IFC-CityGML Unidirectional Conversion. International Journal of Advanced Computer Science and Applications, 3(5). https://doi.org/10.14569/IJACSA.2012.030525

GIS and BIM Integration Will Transform Infrastructure Design, 2018: Redshift EN. https://www.autodesk.com/redshift/gis-and-bim-integration/

Gröger, G., Kolbe, T. H., Nagel, C., & Häfele, K.-H., 2012: OGC City Geography Markup Language (CityGML) Encoding Standard. 344.

Hütter, J., 2016: KIT - IAI - FZKViewer [Text]. https://www.iai.kit.edu/1648.php

Kolbe, T. H., 2009: Representing and Exchanging 3D City Models with CityGML. In J. Lee & S. Zlatanova (Eds.), 3D Geo-Information Sciences (pp. 15–31). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-87395-2_2

Kolbe, 2009: Representing and Exchanging 3D City Models with Ci.pdf. (n.d.).

Lee, N., & Hollar, D. A., 2013: Probing BIM Education in Construction Engineering and Management Programs Using Industry Perceptions. 8.

Arroyo Ohori, K., Diakité, A., Ledoux, H., Stoter, J., & Krijnen, T., 2018: Final report 10 January 2018. 30.

STEP-file, ISO 10303-21., 2017: [Web page]. https://www.loc.gov/preservation/digital/formats/fdd/fdd000448.shtml