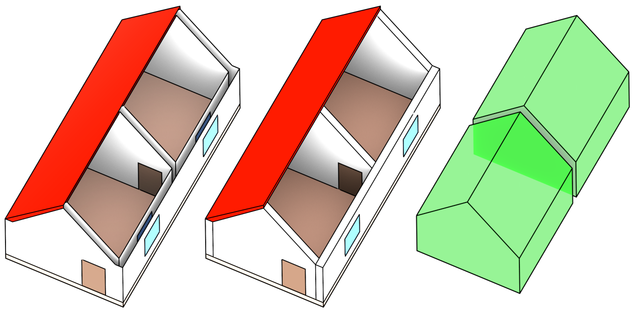
GeoBIM project

Final report

10 January, 2018



Ken Arroyo Ohori
Abdoulaye Diakité
Hugo Ledoux
Jantien Stoter

Thomas Krijnen

Funding organisations:

Project participants:

Delft University of Technology	Ken Arroyo Ohori
	Abdoulaye Diakité
	Hugo Ledoux
	Jantien Stoter
Eindhoven University of Technology	Thomas Krijnen
City of The Hague	Edward de Wit
	Jeroen Schilleman
City of Rotterdam	Louis Smit
Joris Goos	
Rijkswaterstaat	Herman Winkels
Kadaster	Marc Post
	Jantien Stoter
BIM Loket	Dik Spekkink
Geonovum	Friso Penninga
	Jantien Stoter

Contents

1 Motivation 4

2 CityGML and IFC 6
2.1 CityGML 6
2.2 IFC 7

3 Previous integration efforts 8

4 The GeoBIM project 11
4.1 Sample data 11
4.2 Initial methodology 13
4.3 Partial results and issues identified with the initial methodology 14
4.4 Final methodology 18
4.5 Partial results and issues identified with the final methodology . 19

5 A proposed set of IFC modelling guidelines 20
5.1 Other sources of guidelines 21
5.2 Georeferencing 21
5.3 Valid volumetric objects 23
5.4 No self-intersections or intersections between objects 24
5.5 Forming enclosed spaces that are also modelled as IfcSpaces . . . 25
5.6 Consistent use of the most specific entities possible 25

6 Conclusions 26

1 Motivation

Geographic information systems (GIS) have long been used to model the environment and to perform 2D spatial analyses of large areas. However, with the increasing availability of computing power, advanced data acquisition methods, and automated workflows that generate detailed 3D data, GIS models have become increasingly detailed and started to contain models of individual buildings—the traditional domain of building information modelling (BIM).

At the same time, the increase in computing power and the availability of better software have enabled BIM methodologies to move to the mainstream, disrupting more traditional building design platforms based on 2D CAD drawings. As users of BIM software want to incorporate the surrounding features into their workflow, it is only logical that the BIM domain is currently enhancing its standards to support environmental information such as infrastructure and that BIM users turn to existing GIS datasets containing environmental information. Both domains are thus now overlapping, increasingly modelling the same objects, even if the data is represented and stored in rather different ways.

While the GIS and the BIM domains clearly overlap when the modelling of cities is concerned, each domain retains its own focus and has its own characteristics. The BIM domain focuses on information about the design and construction of building sites, and thus has very detailed and semantically rich information about all the physical elements that comprise an individual building as it is designed or built. Meanwhile, GIS describe information about the environment as captured at different points in time, thus having less detailed but regularly updated datasets covering wide regions.

Due to the overlap in the features modelled in both domains as well as their differing strengths and weaknesses, it is widely acknowledged that the integration of data from both domains is beneficial and a crucial step forward for future 3D city modelling. This integration can avoid unnecessary efforts in redundant modelling and allow for new data flows in both directions and new applications. In this way, more detailed BIM data can feed more general GIS data and GIS data can provide the context that is usually missing in BIM data. By pursuing the integration of GIS and BIM data, many new possibilities appear: with contextual GIS information, BIM methodologies can be better applied to infrastructural works; more detailed 3D city models can be built by reusing BIM data; smart city concepts can perform integrated reasoning on terrain, buildings and city infrastructure; and spatial analyses can support multiple levels of detail and the complete life cycles of objects.

Yet, the disciplines of GIS and BIM are disconnected by their modelling paradigms, software tools and open standards—respectively CityGML for GIS (Section 2.1) and IFC for BIM (Section 2.2). Consequently, GIS and BIM datasets differ fundamentally with respect to their semantics, geometry and level of detail, and because of the different modelling approach of both, there is not one optimal nor uniform conversion between the information models. Even as researchers and practitioners have studied how to best share information between BIM and GIS and how to address all the differences from different perspectives (Section 3), it is still very hard (if not impossible) to share 3D information among different users throughout the life cycle of urban and environmental processes, i.e. from plan, design and construction to maintenance. Moreover, most of the research so far has focused on the semantic aspects of GIS-BIM integration (e.g. mapping equivalent types), leaving the difficult task of geometric processing on the background.

In view of these integration problems, in the beginning of 2017, we started the GeoBIM project[1] (Section 4). The project is a collaboration of two research groups on BIM and 3D GIS (Technical universities of Eindhoven respectively Delft), the two respective national standardisation bodies (BIM Loket and Geonovum) and several users who have a high interest in closer BIM/GIS integration, i.e. Rijkswaterstaat, Kadaster and the cities of Den Haag and Rotterdam.

In this project we planned to work on methodologies to process complex IFC and CityGML models concurrently and in an automated fashion. However, since complex architectural models are available mostly in IFC, we prioritised using these models as our input with the aim of creating an interface that could do operations such as performing automated tests on them and converting them to CityGML LOD 3. We created such an interface based on IfcOpenShell[2] and CGAL[3], a pair of libraries respectively used to process BIM and GIS models. If time allowed, we planned to later work on processing complex CityGML models, such as by performing automated conversions to IFC. However, this was not possible given the time constraints of the project.

In this report we present the results of the project, which are mixed. On one hand, we found a series of errors that seem to be pervasive in IFC models and which make automated processing of complex architectural models very difficult, and since fully dealing with such errors would require automatic repair algorithms, a complete working interface could unfortunately not be developed within the timeframe of the GeoBIM project. However, we took the opportunity to look into these errors and converted them into a set of recommended guidelines that should enable the automated processing of IFC models (Section 5). After a further explanation of the two standards CityGML and IFC in Section 2, we describe the objectives, methodology and findings of our GeoBIM project in Section 4. In Section 5 we describe the iFC modelling guidelines to

[1]`https://3d.bk.tudelft.nl/projects/geobim/`

[2]`http://ifcopenshell.org`

[3]`http://www.cgal.org`

better facilitate automated conversion from IFC to CityGML. We finish with our conclusions from this project in Section 6.

2 CityGML and IFC

Due to a wider availability of information, ease of analysis and for pragmatic reasons, the studies on the exchange of BIM and GIS data often focus on the two most prominent open standards in the two domains: the OGC standard *CityGML* for the (3D) GIS domain [Open Geospatial Consortium, 2012] (Section 2.1), and the *Industry Foundation Classes* (IFC) for the BIM domain [ISO, 2013; Building SMART International, 2013] (Section 2.2). IFC models represent the physical elements of single constructions in great detail, while CityGML models represent entire cities in a simpler format that is usable for exchange, dissemination and spatial analyses, such as solar potential and energy consumption estimations. The two modelling paradigms embodied by IFC and CityGML are representative of BIM and 3D GIS data in general, and they are both widely used in their respective domains.

2.1 CityGML

CityGML [Open Geospatial Consortium, 2012] is the most prominent standard to store and exchange 3D city models with semantics in the GIS domain. It presents a structured way to describe the geometry and semantics of topographic features such as buildings and roads. CityGML as a data format is implemented as an application schema for the Geography Markup Language (GML)[4] [OGC, 2004].

CityGML contains a small number of classes structured into 12 modules, most of which are meant to model different types of objects (e.g. Building, Bridge, WaterBody). These classes differ in the way objects are structured into smaller parts and the attributes that are expected for each. However, CityGML geometries are essentially the same for all classes: objects are represented as surfaces embedded in 3D and consist of triangular and polygonal faces.

CityGML supports five levels of detail (LODs). Figure 1 illustrates the five LODs (LOD0 to LOD4) for the building object:

LOD0 is non-volumetric and is an horizontal footprint and/or roof surface representation for buildings;

LOD1 is a block-shaped model of a building (with an horizontal roof);

[4]CityGML uses version 3.1.1 of GML

LOD2 adds a generalised roof and installations such as balconies;

LOD3 adds, among others, windows, doors, and a full architectural exterior;

LOD4 models the interior of the building, potentially with pieces of furniture (CityGML does not mandate which indoor features need to be modelled, in practice resulting in models with a different granularity [Goetz, 2013; Boeters et al., 2015]).

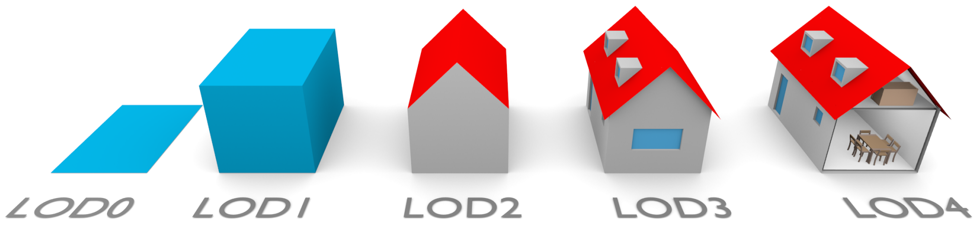


Figure 1: A building represented in LOD0 to LOD4 (image from Biljecki et al. [2016b]).

2.2 IFC

The Industry Foundation Classes (IFC)[5] standard is an open data model used in the Building-information modelling (BIM) domain for the exchange of construction models, often including 3D models of buildings. It has also been adapted as the ISO 16739 international standard [ISO, 2013]. Its geometric aspects are however mostly defined or derived from a different standard, ISO 10303 [ISO, 2014], which also specifies the STEP Physical File (SPF) encoding that is most commonly used in IFC files (`.ifc`).

IFC files can contain many types of classes (130 defined types, 207 enumeration types, 60 select types, 776 entities, 47 functions, and 2 rules in IFC 4 Addendum 2). The geometries in them can use several different representation paradigms which can be combined freely. In practice, most IFC objects are built using sweep volumes, explicit faceted surface models and CSG [El-Mekawy and Östman, 2010]. The representation paradigms include:

- **Primitive instancing:** an object is represented based on a set number of predefined parameters. IFC uses this paradigm to define various forms of 2D profiles (Figure 2), as well as volumetric objects such as spheres, cones and pyramids.

[5] `http://www.buildingsmart-tech.org/specifications/ifc-releases`

- **CSG:** an object is represented as a tree of Boolean set operations (union, intersection and difference) of volumetric objects (see Requicha [1982] for more details). Half-spaces are often used to cut out the undesired parts of surfaces or volumes.
- **Sweep volumes:** a solid can also be defined by a 2D profile (a circle, a rectangle or an arbitrary polygon with or without holes) and a curve [Wang and Wang, 1986] along which the surface is extruded.
- **B-rep:** an object is represented by its bounding surfaces, either triangulated meshes, polygonal meshes or topological arrangements of free-form surfaces.

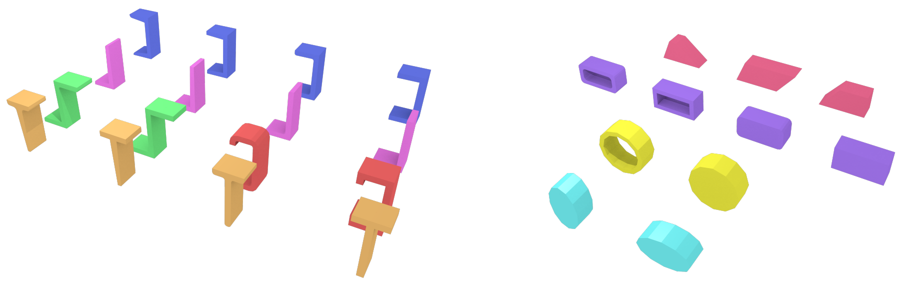


Figure 2: IFC defines various types of parametric curved profiles such as (a) those based on the characters U, L, Z, C and T and (b) those based on trapezia, (rounded) rectangles, circles with/without holes and ellipses. Note the various types of tapered and curved parts of the profiles. These are most commonly used in extrusions such as those shown here.

3 Previous integration efforts

The integration of BIM and GIS data is a complex topic that has been tackled in different ways. As Liu et al. [2017] state, the integration can involve semantics and/or geometry, and it can involve the conversion of BIM and GIS data to either a unified model or to the standards of each other (either bidirectional or only one-way). In the GeoBIM project, we instead focused on realising the integration in practice and at the *data* level. By contrast, Amirebrahimi et al. [2016] identifies how the integration can also be performed at the *process* and *application* levels.

Most of the related work on this topic is concerned with converting IFC models to a GIS model like CityGML because that implies simplifying and removing details and extraneous information in the data. The inverse operation, from GIS data to BIM, is rarely discussed as it is deemed less useful and it might involve adding more details to the data. Nonetheless, there are methods to create BIM from existing models, which can be considered as GIS to BIM; see Volk et al. [2014] for an overview. One use case for the conversion from GIS to BIM is to be able to import GIS data about the environment into BIM software so that designers can consider the environment in their design. An example is converting data about the geological subsurface (defined in GIS formats) into IFC [Diakité and Stoter, 2017].

El-Mekawy et al. [2011] and El-Mekawy et al. [2012] propose to combine information from both domains and create a unified model in which all the semantic properties of IFC and CityGML are present, and they propose using bidirectional mappings for all the semantic classes relevant to IFC and CityGML. However, they extract the geometries separately from the two models and need manual editing to enrich the result. The resulting unified model does not acknowledge the particularities of the two different communities, and their focus is mostly LOD4 since all the features are mapped to LOD4. Amirebrahimi et al. [2016] extends the data model of GML (and not CityGML) to create a unified model supporting two specific applications (visualisation and flood damage assessment to specific buildings).

When the geometry is considered, most existing conversion algorithms from BIM to GIS convert *all* the geometries (from one of the 4 paradigms listed in Section 2.2), which yields GIS models having poor usability in practice. Using these geometries in software for simulations or spatial analyses requires a huge amount of manual work [McKenney, 1998]. Little attention has been paid to a more meaningful conversion which requires not only selecting the appropriate classes, but also performing spatial operations (such as Boolean operations) to select only the features, or part of these, that are appropriate in the other model. One concrete example is how in CityGML the buildings in LOD2 and LOD3 should be modelled using only their surfaces that are visible from the outside, while a simple conversion of all the BIM classes (in most cases) will yield walls that have a thickness and thus interior and exterior surfaces. As Benner et al. [2005] remarks, the surfaces forming the exterior of a building are not explicitly marked as such in an IFC file and cannot be deducted directly from the semantics of the objects.

A few programs offer the possibility to convert IFC models into CityGML models. Three examples are: the Building Information Modelserver[6], IfcExplorer[7] and Safe FME[8]. All of them allow the users to convert IFC models to CityGML at different LODs. The users can in some cases choose which IFC objects should

[6]`http://bimserver.org`
[7]`http://iai-typo3.iai.fzk.de/www-extern/index.php?id=1566&L=1`
[8]`http://www.safe.com`

be used. However, all the features are converted, without any selection or post-processing to keep only the relevant ones. Different projects use such approach to obtain integrated datasets useful for visualisation-based analysis, e.g. Rafiee et al. [2014]. Observe here that for visualisation converting all the geometries is not a major hindrance since the ones that are indoor will simply be not displayed. The size of the dataset will however increase and slow down the visualisation process.

Hijazi et al. [2010] built an open-source web-GIS in which IFC objects can be imported after having been converted to b-rep models. Unfortunately, only sweep-volume objects can be converted and all the geometries are kept (thus resulting in non-manifold building models). De Laat and van Berlo [2011] developed an CityGML ADE (application domain extension) called GeoBIM[9], so that new semantic classes defined in IFC are added in a CityGML model. However, no geometric manipulation is performed.

There have been different attempts at converting IFC models to CityGML LOD2/3 models by processing the geometries. Benner et al. [2005] describe the general steps needed to convert an IFC file to an alternative data model closely related to CityGML (the QUASY model). They first map the semantics from IFC to QUASY and select the relevant boundary objects, and then the outer visible surfaces are extracted by selecting a subset of the input objects. For the (equivalent of) the LOD3 model, they discard geometries inside the building by projecting each floor of a building to horizontal and vertical 'footprints' and keeping only those touching the envelope. This technique may yield building models having holes/gaps in the exterior envelope. Moreover, while the output models appear to be LOD3 models, the walls and the roof are volumetric. Deng et al. [2016] converts IFC to the different LODs of CityGML. To obtain the exterior envelope of each building, they use a ray-tracing algorithm: they define a few points-of-view and determine whether a given surface is visible from them. If so, it is assumed to form part of the exterior boundary. This method will however yield buildings with several holes as several surfaces (e.g. those for a roof overhang, or small ones near a window sill) will not be visible from the finite set of points of view. Donkers et al. [2016] convert IFC models to LOD3 models by selecting a subset of the objects and then extracting the exterior envelope by using a series of Boolean set operations in 3D. Their algorithm does not yield holes/gaps if the input does not contain any and they can close small gaps by buffering all primitives; this however introduces artefacts in the whole model. However, in their implementation the semantics of the objects cannot be stored and thus different tricks are used to imply it in the final model.

Recently, the Open Geospatial Consortium carried out a research on the use of IFC and CityGML in Urban Planning [Kalantari, 2017]. This project was done in the context of the Future City Pilot Phase 1. Their main conclusions align

[9]http://www.citygmlwiki.org/index.php/CityGML_GeoBIM_ADE

to our findings. They firstly conclude that the architecture design processes often do not require georeferencing, which seriously hinders re-use of the data in GIS environments. Instead, the geographical coordinate system of IFC file should be set in advance. Secondly, the OGC project identified several inconsistencies in coding IFC elements that made transformation to CityGML complicated. Therefore they conclude that for adopting IFC in urban planning a clear set of specification needs to be set for the preparation of IFC files. This is line with our findings (see further). However, we go one step further and actually propose such guidelines from our findings in Section 5.

4 The GeoBIM project

To make further steps in the BIM and GIS integration, we started a joint research project called *GeoBIM*. For a successful integration of GIS and BIM data in the project, we considered that two aspects are important. First, data experts from both domains should be involved in the integration endeavour in order to deeply understand the integration issues—most researchers are experts in one of the domains; rarely in both. The second aspect is to have a close involvement of users and use cases to assure that the solution is more than an academic or theoretical standardisation exercise. Applications from practice should define which BIM data is exactly needed in GIS applications and vice versa, and the integration should be implemented accordingly.

The partners in the project are: GIS/CityGML experts from TU Delft, BIM/IFC experts from TU Eindhoven, Geonovum as the Dutch national organisation for GIS data standardisation, BIM-loket as the Dutch national organisation for BIM standardisation, and several users: the Cities of Rotterdam and The Hague as well as Kadaster and *Rijkswaterstaat*, the Dutch government entity responsible for public works and water management.

The GeoBIM project is an experiment- and use case-driven scoping study and has two aims: (1) to develop a CityGML/IFC interface for reusing GIS data in the BIM domain and vice versa and (2) to formulate recommendations for further integration, such as modelling guidelines for bidirectional integration based on the main issues identified and the preferred solutions to these.

4.1 Sample data

We have focused our experiments on three IFC files from the City of The Hague (Figures 3–5). These are complex models with several thousand objects each and which use all the main representation paradigms that are possible in IFC.

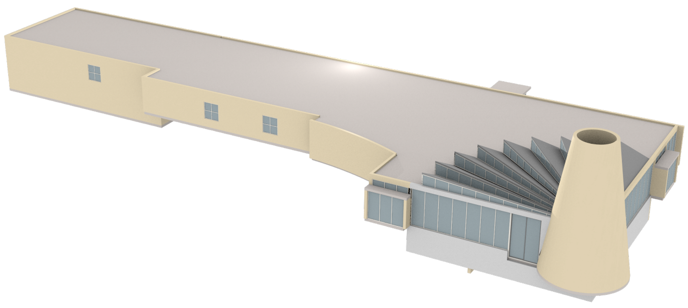


Figure 3: CUVO Ockenburghstraat KOW

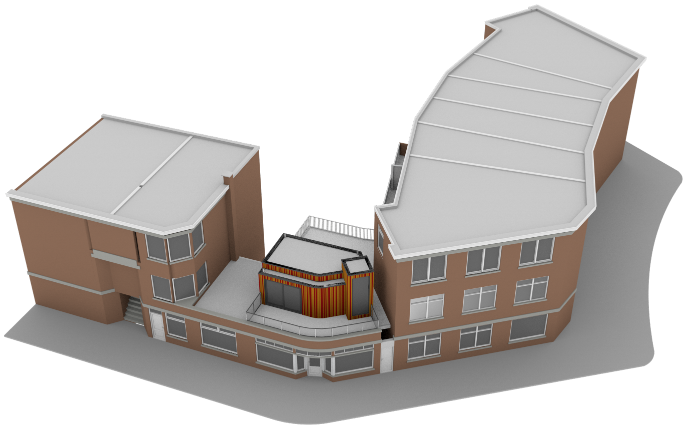


Figure 4: Rabarberstraat 144

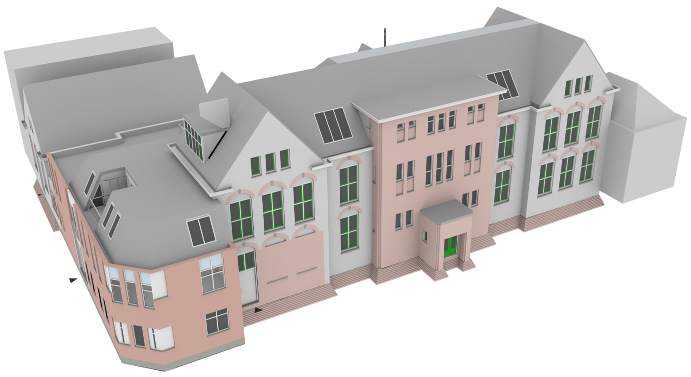


Figure 5: Witte de Withstraat

4.2 Initial methodology

Initially, we attempted to parse and process every object in an IFC file independently using a modified version of IfcOpenShell[10], where we substituted its Open CASCADE[11]-based kernel for a new one based on CGAL (Figure 6). The aim of this approach was to use spatial analysis algorithms from the GIS domain and the robust Boolean set operations on Nef polyhedra available in CGAL in order to solve various use cases [Bieri and Nef, 1988; Nef, 1978; Hachenberger, 2006]. We chose for this approach since our previous experience showed that the Boolean set operations in Open CASCADE are not as robust as those available in CGAL (Figure 7). For instance, compliance with height regulations in 3D zoning maps can be checked using Boolean intersections of the model and the polyhedra generated from zoning maps.

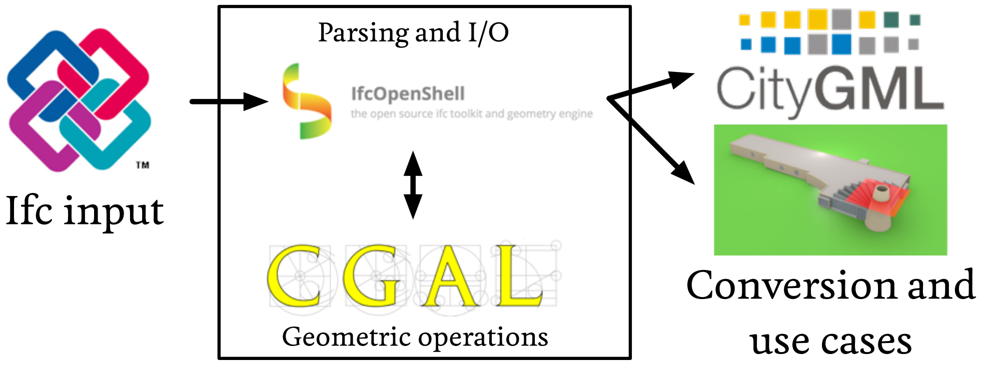
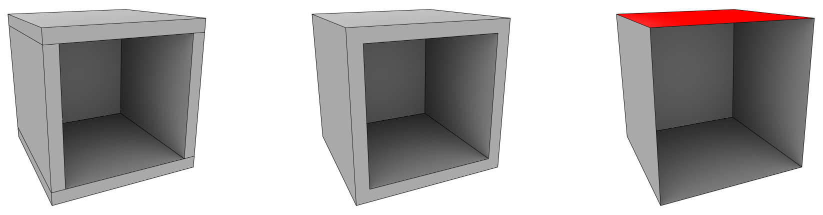


Figure 6: The initial methodology

The main IFC entity types were thus converted into appropriate CGAL-based representations for points, polygonal curves, polygons with holes, planes, etc. In this process, implicit and parametric curves, surfaces and volumes were interpreted into explicit boundary representations and discretised into polygonal curves and polygonal meshes (Figure 2). All placements and transformations in IFC are converted into 3D affine transformations defined by a matrix, which can then be recursively applied to each object as necessary. In this manner, we obtained a polyhedral representation of every volumetric object and store it as a CGAL `Polyhedron_3`.

Some IFC entity types however required additional processing, such as the CSG solid, half-space and Boolean result representations which are obtained by first converting a polyhedron to a CGAL `Nef_polyhedron_3` and then performing Boolean set union, intersection or difference operations.

[10] http://ifcopenshell.org
[11] https://www.opencascade.com



(a) (b) (c)

Figure 7: A method to get the semantically labelled outer surfaces of (a) a simple IFC model in a manner suitable for CityGML LOD3 could involve (b) computing the Boolean set union of certain IFC entities (e.g. IfcSlabs) and (c) extracting the outer envelope and classifying surfaces using their normals [Boeters et al., 2015; Donkers et al., 2016].

4.3 Partial results and issues identified with the initial methodology

This methodology yielded good initial results (Figures 8–10), being able to parse and generate CGAL geometries for a rapidly increasing number of the objects present in our test IFC models.

However, we found that invalid objects are widespread in the IFC models we received as part of this study—something that is consistent with our previous experience with BIM and GIS data and with the experience of other practitioners we spoke to; see for instance Biljecki et al. [2016a]. Self-intersections and intersections between objects were the most common errors (Figures 11 and 12), but there are also uneven surfaces that are supposed to be planar and disconnected objects that are modelled as one. One of the most interesting errors we found were objects that are seemingly valid and form topological 2-manifolds (as checked by making sure that surfaces join in pairs along common edges), but in reality contain self-intersections (Figure 13). Note that this is something explicitly disallowed by the IFC standard[12], but not enforced by most current implementations.

Since invalid objects would often cause our processing methods to crash, we attempted to catch and correct as many errors as possible, and so we added a series of validation tests on every object and to its openings (if any). These are done at appropriate places during the construction of every object, before the

[12]http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcgeometricmodelresource/lexical/ifcfacebasedsurfacemodel.htm

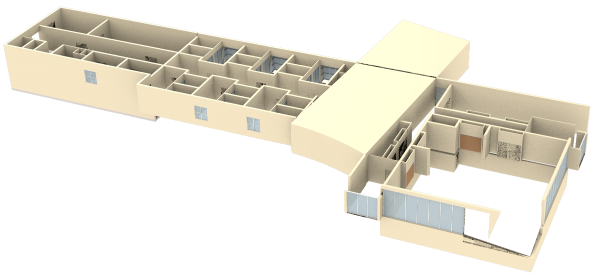


Figure 8: CUVO Ockenburghstraat KOW

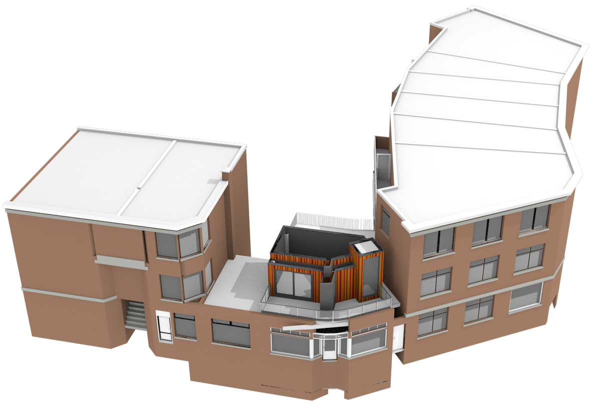


Figure 9: Rabarberstraat 144

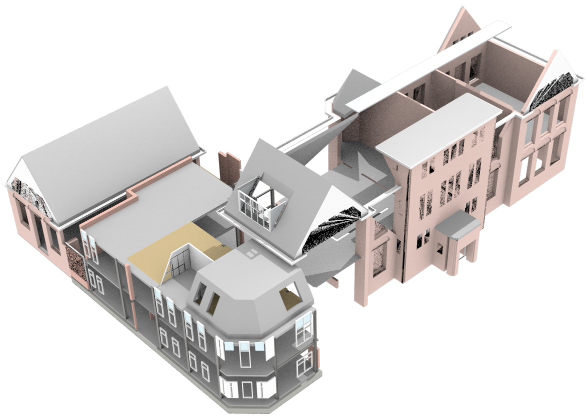


Figure 10: Witte de Withstraat

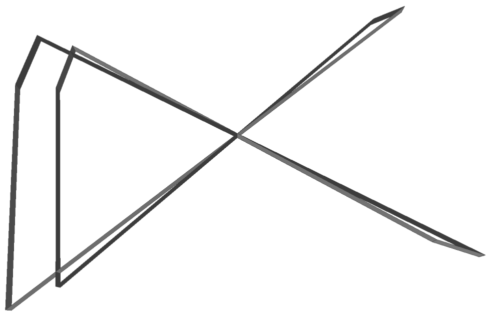


Figure 11: A prismatic polyhedron with an obvious self-intersection. The self-intersecting top and bottom faces of the polyhedron are not shown.

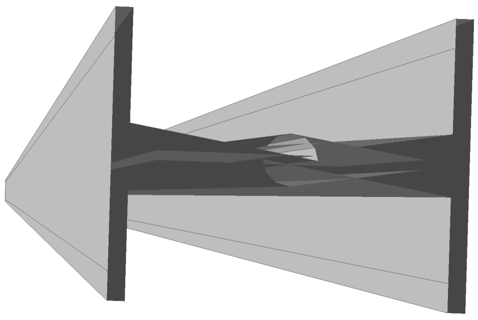


Figure 12: A self-intersecting representation of a beam.

conversion of a CGAL `Polyhedron_3` to a `Nef_polyhedron_3`, and before it is triangulated for the generation of a file used for visualisation (the simple OBJ format). Among others, we tested for:

- combinatorial validity (2-manifoldness),
- surfaces that enclose a space,
- crashes/failures of CGAL's triangulation algorithm (e.g. when a surface self-intersects),
- self-intersections,
- CGAL crashes/failures when converting to a `Nef_polyhedron_3`.

In order to make sure that every face is perfectly planar, we also triangulated the non-triangular faces of every object whenever we need to create a Nef polyhedron. This ensured that the conversion from a CGAL `Polyhedron_3` to a

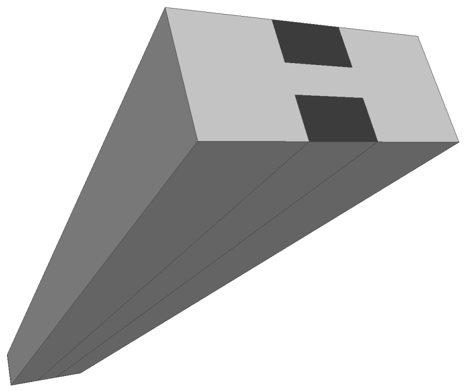


Figure 13: A topological manifold that contains non-obvious geometric intersections. The bottom of the polyhedron, seemingly composed of three rectangular faces, actually has only two rectangular faces that overlap along the middle third. The top of the polyhedron (not visible) has the same problem. Typical validation tests commonly return that such a shape is valid.

`Nef_polyhedron_3` is able to compute a plane passing through every face. Another possibility would have been to compute the best fitting plane per face and then snap a vertex to the intersection of the planes of its incident faces [Arroyo Ohori, 2016].

An issue we identified is the potential for differing results in the generation of discrete and explicit b-rep linear geometries (e.g. polygons and polygonal meshes) from the implicit and curved geometries in IFC. Explicit geometries will invariably yield different b-rep representations according to the chosen discretisation method and its parameters. For instance, we discretise ellipses into closed polygonal curves using a customisable number of equal-angle intervals, and we discretise spheres into icosahedral approximations with a customisable number of refinements. However, alternative methods could involve discretising ellipses using equal-length line segments and spheres using equal angle rectangular patches.

From a more practical standpoint, we also found that the available features of CGAL are not enough to comfortably model all the complex features of IFC—something that is also true for many other geometric processing libraries. For instance, CGAL Nef polyhedra do have support for half-space representations as long as an extended kernel is used, which incorporates polynomial representations of various classes such as planes. However, extended CGAL kernels appear to be incompatible with various parts of the Polygon mesh processing package, which we use for triangle-triangle intersection tests, 2-manifoldness tests, stitching the faces of a polyhedron together, and reversing normals among other functions. Another example is that Nef polyhedra do not offer a quick way to construct meshes that do not enclose a volume or to create a `Polyhedron_3` from such a mesh stored in a Nef polyhedron. While we have found (and partly implemented) workarounds around these CGAL problems, these involve very complex code, are too slow for practical use or do not cover all the many cases commonly present in IFC files.

Unfortunately, while trying to fix all the errors present in the model, it thus became clear that getting all IFC entities to work correctly using this conversion methodology was going to be a significantly more time-consuming undertaking than the project's allotted time allowed.

4.4 Final methodology

After running against the time constraints for the project and due to the unsolvable issues identified with the initial methodology, we applied a simplified process that minimised the complex interaction between CGAL and IfcOpenShell (Figure 14). The idea was to sidestep the limitations of CGAL in representing certain geometries and avoid many of the CGAL crashes caused by the interplay between invalid geometries in the IFC files and the stricter geometry requirements of CGAL compared to Open CASCADE.

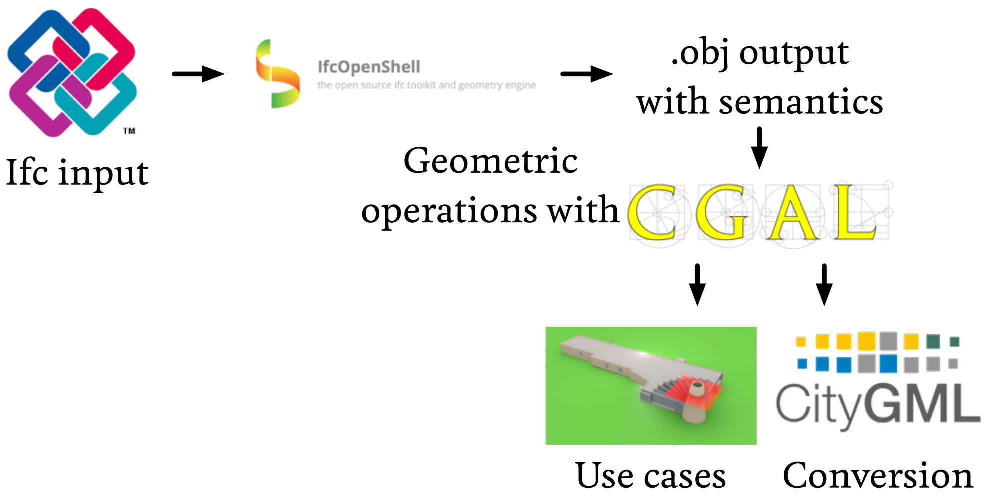


Figure 14: The final methodology

In this final methodology, we wrote a script that used the standard version of IfcOpenShell (with its Open CASCADE kernel) to export a series of Wavefront .obj files containing all relevant objects present in the IFC files. For this, we mainly looked at the subentities of IfcBuildingElement, which according to the IFC standard define a 'major functional part of a building', those being: IfcBeam, IfcBuildingElementComponent, IfcBuildingElementProxy, IfcChimney, IfcColumn, IfcCovering, IfcCurtainWall, IfcDoor, IfcFooting, IfcMember, IfcPile, IfcPlate, IfcRailing, IfcRamp, IfcRampFlight, IfcRoof, IfcShadingDevice, IfcSlab, IfcStair, IfcStairFlight, IfcWall, IfcWindow. Thus, all geometries in every IFC file were extracted to an easily-parsable format while preserving the most relevant semantic information present in the original input file.

Afterwards, we created a different program that parsed the individual .obj files and created a CGAL Nef polyhedron for every object. The parsed objects were processed using the same defensive programming methods and validation tests described for the initial methodology. The obtained Nef polyhedra were then selectively processed together using Boolean set operations so as to obtain a single Nef polyhedron containing all of the relevant parts of a building for a given spatial analysis.

4.5 Partial results and issues identified with the final methodology

The simplified processing pipeline in the final methodology resulted in a more workable methodology for the time frame of our project. All objects in our three test IFC datasets were able to be exported correctly to .obj and appeared to be visually correct. Moreover, many of the problems present in the IFC files could indeed be sidestepped by using the Open CASCADE kernel of IfcOpenShell, resulting in a significantly smaller number of problematic objects (e.g. such as a very complex staircase with many self-intersections), which could in general be disregarded for our purposes.

By using Boolean set operations we were also able to correctly handle overlapping objects, and by combining these with Minkowski sums we also found reasonable solutions to many of the remaining problems in the IFC files. For instance, non-manifolds could be converted into manifolds by finding the non-manifold edges and vertices in an object, applying a Minkowski sum of these with a small volumetric kernel, and subtracting the result from the original objects using a Boolean set difference. Similarly, small gaps between objects that would cause the interiors of a space (e.g. a room) not to be enclosed could be fixed by applying a Minkowski sum with a small kernel of a width equivalent to a snapping threshold between adjoining volumes. As another example, some CGAL crashes could be avoided by applying an iterative process where program

Guideline	Purpose
Georeferencing	Link IFC model to real-world coordinates
Valid volumetric objects	Create processable objects that enclose a space
No intersections	Avoid ambiguity and processing problems
Enclosed spaces	Unambiguous definitions of empty spaces
Consistent specific entities	Ease of processing and conversion

Table 1: Summary of our proposed IFC modelling guidelines

crashes are caught, and an object that otherwise causes a crash is processed by applying a Minkowski sum with a small kernel until the object can be handled properly.

Unfortunately, even as managed to process significantly more objects using the final methodology, we were still not able to fix all outstanding issues in the time allotted for the project, and thus were not able to process all objects in the IFC files. This is a problem, since the automated tests we considered (e.g. zoning and shadow casting) cannot yield authoritative results if the models have missing objects. Similarly, meaningful conversions from IFC to CityGML cannot be performed when spaces that should be well enclosed are not, due to missing objects at their boundaries.

5 A proposed set of IFC modelling guidelines

The development process of the two methodologies pointed to a series of issues that are common in IFC files, including in the files we received for this project. We thus looked further into these issues and produced a set of guidelines to avoid them, which are summarised in Table 1. If properly followed, they would greatly simplify the automated processing of IFC files.

Many rules and recommendations regarding the proper use of IFC are already given in the IFC standard, implementation guidance, and external guidelines, among others. These range from fundamental aspects, such as how each IFC entity is defined and the possible values for each attribute, to common-sense practical rules, such as schemes for the consistent naming of objects. We make a brief summary of what is specified in each of these in Section 5.1.

However, during the course of this project and based on previous experiences, we have found that the above mentioned sources of rules and recommendations are not always followed fully and do not cover certain desirable aspects, resulting in a series of issues that make processing IFC models less than ideal. We therefore propose in Sections 5.2–5.6 a series of recommendations that aim to improve their reusability in Geo applications.

5.1 Other sources of guidelines

The official **IFC standard**[13] is the main source of IFC documentation. It defines the fundamental concepts and assumptions underlying the standard, such as a project (i.e. a structure containing the context and objects of a model), the object definition and association (i.e. links to other sources), product and product type shapes (i.e. the geometry and topology of objects) and composition (i.e. hierarchies of objects).

Most importantly, the standard defines all IFC entities and their attributes, which are specified in a set of separate schemas. These form a hierarchy, which includes core schemas (which are used throughout the standard), shared schemas (which are not in the core, but are still used in other parts of the standard), and domain schemas (which are self-contained and based on a specific field). In addition, there are resource schemas that define supporting data structures that can be used by all IFC entities.

buildingSMART provides a series of resources that are meant as **implementation guidance** for IFC2x3[14] and IFC4[15]. These include an implementers guide for IFC2x3[16], which describes in detail the logic behind the main entities of IFC2x3 and how they can be implemented, a series of implementer agreements[17], which cover aspects that can be unclear in the standard, and links to useful tools and examples.

External sets of guidelines for IFC are also available, such as the **BIM Basic IDM**[18] (*BIM basis ILS*). Among other aspects, this document recommends the use of a certain format for names, the correct use of entities, avoiding duplicate or intersecting objects, and filling in the values for certain important properties.

5.2 Georeferencing

Properly georeferencing an IFC file makes it possible to link the coordinates inside an IFC model with its real-world coordinates and thus to place the model of a single building or construction within the virtual environment. For this,

[13]`http://www.buildingsmart-tech.org/ifc/IFC4/final/html/`
[14]`http://www.buildingsmart-tech.org/implementation/ifc-implementation`
[15]`http://www.buildingsmart-tech.org/implementation/ifc4-implementation`
[16]`http://www.buildingsmart-tech.org/downloads/accompanying-documents/guidelines/IFC2xModelImplementationGuideV2-0b.pdf`
[17]`http://www.buildingsmart-tech.org/implementation/ifc-implementation/ifc-impl-agreements`
[18]`http://bimloket.nl/BIMbasisILS`

it is possible to use the IFC entity `IfcSite`, which defines an area where construction works are undertaken, and optionally allows for storage of the real-world location of a project using 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 world geodetic system WGS84 (EPSG:4326). Positive values represent locations north of the equator, west of the geodetic zero meridian (nominally the Greenwich prime meridian) in IFC2x3, or east of the zero meridian IFC4. Negative values represent locations south of the equator, east of the zero meridian in IFC2x3, or west of the zero meridian in IFC4. These angles are expressed according to the type `IfcCompoundPlaneAngleMeasure` and all components (i.e. degrees, minutes, seconds and millionth-seconds of arc) should have the same sign. According to the IFC standard, the geographic reference given might be the exact location of the origin of the local placement of the IfcSite or it might be an approximate position for informational purposes only. The elevation is defined according to the datum elevation relative to sea level.

In addition, `IfcSite` contains a couple of other attributes that allow for an approximation of the real-world location to be given. The *LandTitleNumber* can store the designation of the site within a regional system (e.g. a cadastral record ID), and the *SiteAddress* can store the postal address of the site.

The IFC entity `IfcGeometricRepresentationContext` is used to define the coordinate space of an IFC model in 3D and optionally the 2D plan of such a model. This entity can be used to offset the project coordinate system from the global point of origin using the *WorldCoordinateSystem* attribute, it defines the *Precision* under which two given points are still assumed to be identical, and it defines the direction of the *TrueNorth* relative to the underlying coordinate system. The latter attribute defaults to the positive direction of the y-axis of the *WorldCoordinateSystem*.

In theory, the use of the latitude, longitude and altitude values in the `IfcSite` with an optional offset and true North direction given by the `IfcGeometricRepresentationContext`, should make it possible to precisely georeference an IFC model. In fact, in our experience most IFC files do fill in the requisite values in the IfcSite. However, these values are almost always set to zero, to a completely wrong location or to a very rough approximation of the real location (e.g. a point in the same city). This is unfortunately compounded by the mismatched definitions of the positive direction for the longitude in IFC2x3 and IFC4.

We therefore recommend that IFC files set their precise real-world location using the latitude, longitude and altitude values in the `IfcSite` taking into account the offset given by the *WorldCoordinateSystem* of the `IfcGeometricRepresentationContext` for the 3D model and the 2D plan (if used). Due to practical difficulties, it cannot be expected that these values match the reality with the *Precision* given in the `IfcGeometricRepresentationContext`, but the values should be easy to set to within a few meters of the real location. In addition, if the y-axis of the *WorldCoordinateSystem* in the `IfcGeometricRepresentationContext` does not match the true North direction, the *TrueNorth* attribute should be set as well.

5.3 Valid volumetric objects

There are many possible IFC entities that are subtypes of `IfcRepresentationItem` and can be used to define the geometries of the objects in an IFC model. Given easily verifiable valid attributes (e.g. having a positive value), some of these entities always form valid objects, such as the parametric `IfcBlock`, `IfcBoundingBox`, `IfcRectangularPyramid`, `IfcRightCircularCylinder`, `IfcRightCircularCone` and `IfcSphere`, but most others are prone to various types of problems.

Some of these could be minor and non-consequential, such as a `IfcBooleanClippingResult` or `IfcHalfSpaceSolid` that results in empty space or performs point set intersections with empty spaces. These problems result only in inefficiencies when processing. In our experience, such minor errors are rare in IFC files since it is most often used as an exchange format and such errors are generally not exported even when these operations are performed in software.

Many others of the most commonly used IFC entities are designed in a way that they mostly create valid volumetric objects. The sweeps of `IfcExtrudedAreaSolid`, `IfcRevolvedAreaSolid` and `IfcSurfaceCurveSweptAreaSolid` should create a valid volume given a valid profile definition (i.e. the curve or surface that is swept). This means that the profile should not self-intersect. Somewhat more care needs to be taken with sweeps that allow for arbitrary sweep curves, such as `IfcSectionedSpine` and `IfcSweptDiskSolid`, where the combination of the spine curve and the planar cross-sections can cause the volume to self-intersect.

More common errors are found with entities with a more complex definition, such as with the b-rep creation of shells that self-intersect or do not enclose volumes. This includes other entities such as `IfcFacetedBrep` and `IfcFacetedBrepWithVoids` only enforce the creation of a volumetric object in their definition but can be easily used to create objects that are not closed and thus have no well-defined volume. Note that this is distinct from other entities such as `IfcOpenShell`, which are meant to create non-volumetric objects.

Finally, another common validity constraint is that certain b-rep entities most be composed of smooth 2-manifolds. In particular, `IfcFacetedBrep` and `IfcFacetedBrepWithVoids` require a finite and non-zero extent, non-duplicate faces that are 2-manifold embeddable in the plane within the domain of the face, non-intersecting faces (except at their boundaries), consistent normal orientations, and compliance with a version of the Euler equation. Perhaps most importantly, the manifoldness constraint is defined combinatorially, such that ‘each edge along the boundary of a face is shared by at most one other face in the assemblage’. In our experience, most of these propositions, although prescribed by the standard, are frequently not tested by IFC software. Duplicate and intersecting faces are quite common.

While all of the subtypes of `IfcRepresentationItem` have different uses, we recommend that practitioners use volumetric objects as much as possible, test that they conform to their entity definition and take care to make sure these are watertight when using b-rep representations.

5.4 No self-intersections or intersections between objects

IFC entities generally disallow self-intersections in their definition, such as the examples of `IfcFacetedBrep` and `IfcFacetedBrepWithVoids` presented previously. Among others, `IfcSectionedSpine` states that ‘non [sic] of the cross sections, after being placed by the cross section positions, shall intersect’, `IfcArbitraryClosedProfileDef` that ‘the *OuterCurve* shall not intersect’, and `IfcSweptDiskSolid` that ‘the *Directrix* shall not be based on an intersecting curve’. However, these sort of restrictions are easier to test with some entities than others and are often not enforced in software. Consequently, software can produce invalid IFC models, without any warnings,

To the best of our knowledge, the standard is not explicit about limiting intersections between objects and in practice most IFC files have small intersections between objects—sometimes below the model’s precision but sometimes above. The smallest of such intersections are often caused by numerical errors when adjacent objects are processed separately. Larger intersections can be caused by the manual manipulation of adjacent objects, or deliberately in order to reduce modelling work.

These kinds of intersections might not seem problematic when IFC models are only used for visualisation purposes, but they do harm more complex automated processing efforts. Among other problems, they cause ambiguous models (which can be interpreted differently by different users) and can cause different results when used in different processes. It is worth noting that even small intersections below the precision threshold can cause small problems when processing since this precision can be dealt with in different ways.

We therefore recommend that users ensure that there are no self-intersections or intersections between objects, much like the BIM Basic IDM disallows intersections between objects by stating: ‘There are no duplicates or intersections permitted. Make sure this is checked in IFC.' Among others, the Solibri Model Checker[19] and the simplebim Space Boundary add-on[20] are able to detect overlapping objects.

5.5 Forming enclosed spaces that are also modelled as IfcSpaces

While intersections between objects are undesirable, gaps between objects that are supposed to be adjacent are also a problem. Whereas BIM applications mostly focus on the built space, many Geo applications focus on the empty space between the objects modelled in a BIM file, such as explicit rooms, hallways, stairways, and so on.

Because of the above, we recommend that the necessary objects so as to form enclosed spaces should be modelled, and these objects should fit properly with each other and without gaps between them. In addition, we recommend that enclosed spaces are also modelled explicitly and with their precise geometry as `IfcSpaces`. Simpler applications that make straightforward use of these `IfcSpaces` can then avoid more complex geometry processing.

5.6 Consistent use of the most specific entities possible

IFC provides several entities that are used to mark specific expected elements of a model including the subtypes of `IfcBuildingElement` for buildings (`IfcBeam`, `IfcChimney`, `IfcColumn`, `IfcCovering`, `IfcCurtainWall`, `IfcDoor`, `IfcFooting`, `IfcMember`, `IfcPile`, `IfcPlate`, `IfcRailing`, `IfcRamp`, `IfcRampFlight`, `IfcRoof`, `IfcShadingDevice`, `IfcSlab`, `IfcStair`, `IfcStairFlight`, `IfcWall`, `IfcWindow`), and less importantly `IfcCivilElement`, `IfcDistributionElement`, `IfcElementAssembly`, `IfcElementComponent`, `IfcFeatureElement`, `IfcFurnishingElement`, `IfcGeographicElement`, `IfcTransportElement`, `IfcVirtualElement` and their subtypes.

However, we have found a couple of problems with the use of these entities: (i) they are not always used consistently between models (or even sometimes in a single model), and (ii) they often involve the use of the generic `IfcBuildingElementProxy` rather than one of the other more specific entities. While both of these are minor problems when processing models manually, they make it needlessly difficult to perform automated processing (e.g. deducing the correct entity types from names and descriptions).

We therefore recommend that users take care to always use the most specific entity type possible, to use such entity types consistently across all objects of a

[19]https://www.solibri.com/faq/using-the-space-validation-rule-to-ensure-model-accuracy/
[20]http://datacubist.com/support/addon-spaceboundary.html

model, and to ensure that all related geometries of an object are marked as such. Additionally, when conversions to CityGML are concerned, it is worth focusing on entities that have a direct mapping to CityGML classes, such as `IfcSlab`.

6 Conclusions

Based on the results of the project, it can be concluded that a full integration of GIS and BIM data is far from straightforward given current practice. From a data perspective, this is mainly because existing BIM models contain many geometrical and topological errors which need to be properly handled and often fixed before a conversion can be successful. These may not be problematic when used in a BIM environment because of a few reasons: many more geometry types are usually natively supported in BIM software than in GIS software; geometric errors (e.g. self-intersections) are often found in places that are inconspicuous or invisible from the outside; data is often only used for visualisation purposes, which does not require geometric and topological correctness; or the errors only show after the implicit and parametrised types have been converted into explicit geometries (which is during the conversion to IFC). However, the errors are very problematic in many applications that involve the automated processing of IFC data, such as in GIS operations that perform spatial analyses, which often involve complex operations such as Boolean set operations. Therefore, functionalities to validate the geometry of a BIM model are required; see Ledoux [2013] for an equivalent for 3D GIS datasets.

Another conclusion that can be drawn is that it is unrealistic to develop a robust process for all IFC geometry entity types within a small project. In our experiments, we have so far developed a solution for only a subset of the IFC standard. It would take years to extend our efforts to cover all entities and in practice many of them are rarely used. From a GIS perspective, a related conclusion is that with IFC there are many different ways to model an object, and that supporting all of them is a problem in practice, especially for researchers and small developers, which hinders adoption of the standard and standardisation in the community. Ideally, standards that are intended for exchange should define only one way to model something—this both fosters data exchange and makes it easy to create compliant implementations of the standard. Therefore we recommend to formulate and agree on guidelines to standardise the way a specific situation should be modelled in IFC.

The history of data exchange and its accompanying problems is also much longer in the GIS domain. As also concluded by Liu et al. [2017] from Cerovsek [2011], the relatively younger concept of BIM and its standard IFC have not satisfied the requirements of standards yet: competitiveness, conformity, and connectivity. BIM has to further develop in this respect, including paying attention to formal definitions for correct geometries, adhering to these and developing functionality to validate models.

Many academic research has showed successful IFC-CityGML transformations. However, they mostly develop their own optimal solution, based on the use case or data at hand. A transformation from IFC to CityGML that works in practice, requires a standardised transformation as there are currently many different interpretations of how to best transform an IFC file into CityGML. From a geometric perspective, this diversity in transformation is mainly due to the many classes of IFC that need to be converted into the relatively few classes of CityGML. The processing needed to make explicit geometries from implicit geometries and to change geometry classes (like conversion of volumes in IFC into surfaces in CityGML for walls) can result in different outcomes in different implementations. For a sustainable information chain that supports the life cycle of objects, this is unwanted, and therefore there is a need to define one uniform and standardised transformation.

From our experiments we can also conclude that additional information for building modellers is needed in order to support a better automated processing pipeline for IFC geometries in a GIS environment. We believe that our guidelines are a good first step in this direction, which should help to create CityGML/GIS-ready IFC datasets. However, additional efforts are needed since the export process from other BIM formats to IFC is rather opaque and the implementation of the guidelines is not straightforward.

More understanding is also needed in order to find out how BIM data are used in GIS applications and vice versa. For example, we wonder about how IFC designs are checked against the existing physical world and against a 3D zoning plan (both GIS data): is this done in BIM software, and thus GIS data is imported and localised in BIM software; or is this done in GIS software, and thus BIM design data needs to be imported in GIS software and properly georeferenced.

BIM data is usually much more detailed than what is expected from GIS data. It is thus very unlikely that all the details of a BIM dataset will be integrated into GIS data. Instead, a generalised version of the BIM model (with relevant attributes for the GIS world) will be converted into a GIS model. A 3D city model may serve as a connection between the two, with unique identifiers and update mechanisms in order to keep the separated BIM models consistent with their generalised counterparts.

References

Sam Amirebrahimi, Abbas Rajabifard, Priyan Mendis, and Tuan Ngo. A BIM-GIS integration method in support of the assessment and 3D visualisation of flood damage to a building. 61(2):317–350, April 2016.

Ken Arroyo Ohori. *Higher-dimensional modelling of geographic information*. PhD thesis, Delft University of Technology, apr 2016.

J Benner, A. Geiger, and K. Leinemann. Flexible generation of semantic 3D building models. In G. Gröger and T. H. Kolbe, editors, *Proceedings 1st International Workshop on Next Generation 3D City Models*, pages 17–22, Bonn, Germany, 2005.

H. Bieri and W. Nef. Elementary set operations with *d*-dimensional polyhedra. In Hartmut Noltemeier, editor, *Computational Geometry and its Applications*, volume 333 of *Lecture Notes in Computer Science*, pages 97–112. Springer Berlin Heidelberg, 1988.

Filip Biljecki, Hugo Ledoux, Xin Du, Jantien Stoter, K. H. Soon, and V. H. S. Khoo. The most common geometric and semantic errors in CityGML datasets. volume IV-2/W1 of *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 13–22, Athens, Greece, 2016a. doi: http://dx.doi.org/10.5194/isprs-annals-IV-2-W1-13-2016.

Filip Biljecki, Hugo Ledoux, and Jantien Stoter. An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59:25–37, 2016b.

Roeland Boeters, Ken Arroyo Ohori, Filip Biljecki, and Sisi Zlatanova. Automatically enhancing CityGML LOD2 models with a corresponding indoor geometry. *International Journal of Geographical Information Science*, 29(12):2248–2268, December 2015.

Building SMART International. Industry Foundation Classes (IFC), IFC4, 2013. Available from: http://www.buildingsmart-tech.org/specifications/ifc-releases/summary.

Tomo Cerovsek. A review and outlook for a 'building information model' (BIM): A multi-standpoint framework for technological development. *Advanced Engineering Informatics*, 25:224–244, 2011.

Ruben de Laat and Léon van Berlo. Integration of BIM and GIS: The development of the CityGML GeoBIM extension. In T. H. Kolbe, G. König, and C. Nagel, editors, *Advances in 3D Geo-Information Sciences*, Lecture Notes in Geoinformation and Cartography, pages 211–225. Springer-Verlag, Berlin Heidelberg, 2011.

Yichuan Deng, Jack C P Cheng, and Chimay Anumba. Mapping between BIM and 3D GIS in different levels of detail using schema mediation and instance comparison. *Automation in Construction*, 67:1–21, July 2016.

Abdoulaye Diakité and Jantien Stoter. Eindrapport scoping studie voor integratie geotop en bim: Als input voor de ontwikkeling van basis registratie ondergrond. Technical report, Delft University of Technology, jun 2017.

Sjors Donkers, Hugo Ledoux, Junqiao Zhao, and Jantien Stoter. Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings. *Transactions in GIS*, 20(4):547–569, 2016.

Mohamed El-Mekawy and Anders Östman. Semantic mapping: an ontology engineering method for integrating building models in IFC and CityGML. *Proceedings of the 3rd ISDE Digital Earth Summit*, pages 12–14, 2010.

Mohamed El-Mekawy, Anders Östman, and Khurram Shahzad. Towards interoperating CityGML and IFC building models: A unified model based approach. *Advances in 3D Geo-Information Sciences*, pages 73–93, 2011.

Mohamed El-Mekawy, Anders Ostman, and Ihab Hijazi. A unified building model for 3D urban GIS. *ISPRS International Journal of Geo-Information*, 1:120–145, July 2012.

Marcus Goetz. Towards generating highly detailed 3D CityGML models from OpenStreetMap. *International Journal of Geographical Information Science*, 27 (5):845–865, May 2013.

Peter Hachenberger. *Boolean Operations on 3D Selective Nef Complexes Data Structure, Algorithms, Optimized Implementation, Experiments and Applications.* PhD thesis, Saarland University, 2006.

Ihab Hijazi, Manfred Ehlers, and Sisi Zlatanova. BIM for geo-analysis (BIM4GEOA): set up of 3D information system with open source software and open specifications (OS). In *Proceedings 5th International 3D Geoinfo conference*, pages 45–49, Berlin, Germany, 2010.

ISO. *Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries.* International Organization for Standardization, March 2013.

ISO. *Industrial automation systems and integration - Product data representation and exchange.* International Organization for Standardization, August 2014.

Mohsen Kalantari. Future city pilot-1: Using ifc/citygml in urban planning engineering report. Ogc engineering report, Open Geospatial Consortium, jun 2017.

Hugo Ledoux. On the validation of solids represented with the international standards for geographic information. *Computer-Aided Civil and Infrastructure Engineering*, 28(9):693–706, 2013. ISSN 1467-8667. doi: http://dx.doi.org/10.1111/mice.12043.

Xin Liu, Xiangyu Wang, Graeme Wright, Jack Cheng, Xiao Li, and Rui Liu. A State-of-the-Art Review on the Integration of Building Information Modeling (BIM) and Geographic Information System (GIS). *ISPRS International Journal of Geo-Information*, 6(2):53, February 2017.

Dan McKenney. Model quality: The key to CAD/CAM/CAE interoperability. Technical report, International TechneGroup Incorporated, 1998.

Walter Nef. *Beiträge zur Theorie der Polyeder*. Herbert Lang, Bern, 1978.

OGC. Geography markup language (GML) encoding specification. Open Geospatial Consortium inc., 2004. Document 03-105r1, version 3.1.1.

Open Geospatial Consortium. OGC City Geography Markup Language (CityGML) Encoding Standard 2.0.0. Technical report, April 2012.

Azarakhsh Rafiee, Eduardo Dias, Steven Fruijtier, and Henk Scholten. From BIM to geo-analysis: View coverage and shadow analysis by bim/gis integration. *Procedia Environmental Sciences*, 22:397–402, 2014. ISSN 1878-0296. doi: http://dx.doi.org/10.1016/j.proenv.2014.11.037. URL `http://www.sciencedirect.com/science/article/pii/S1878029614001844`.

A. A. G. Requicha. Representation of rigid solids—theory, methods and systems. *ACM Computing Surveys*, 12(4):437–464, 1982.

Rebekka Volk, Julian Stengel, and Frank Schultmann. Building information modeling (BIM) for existing buildings—literature review and future needs. *Automation in Construction*, 38:109–127, 2014.

WP Wang and KK Wang. Geometric modeling for swept volume of moving solids. *Computer Graphics and Applications, IEEE*, 6(12):8–17, 1986.