Special Issue: Nexus of BIM and GIS: integrating building and GeoSpatial data

A model based approach to convert a building BIM-IFC data set model into CityGML

Kamel Adouane⋆[a], Rudi Stouffs[a], Patrick Janssen[a], Bernd Domer[b], Filip Biljecki[a]

[a] Department of Architecture, School of Design and Environment, National University of Singapore, Singapore, Singapore

[b] HES-SO, University of Applied Sciences and Arts Western Switzerland, Geneva, Switzerland

⋆ Corresponding Author
Accepted 18 August 2019
Proof reading window: 11 September 2019 - 16 September 2019
Online publication: within 10 days after correction reception
Keywords: Spatial, Digital, Building Information Modelling (BIM-IFC); geographical information system (CityGML), BIM GIS convergence

1 Abstract

Digital building modelling faces issues related to inconsistent integration/interoperability, in particular for a BIM GIS convergence. A consistent data conversion approach should consider semantic and spatial geometric modelling, therefore potentially leading to a loss of spatial geometric information. This paper simplifies semantic solving, and concentrates on the accuracy of the spatial geometric representation to avoid conflicting 3D spatial mismatches. To solve the geometry, original spatial specifications in 3D described in the IFC data schema are considered for all objects present in the model. The method is explored with a main BIM-IFC model, and additionally tested with two other models.

2 Introduction

Digital technologies lead towards a mutation of the AEC industry, thanks to the massive collected information available for architects, engineers and constructors. A building's digital representation for spatial physical properties modelling can be obtained through diversified categories of dedicated Building Information modelling (BIM) frameworks. BIM models are notably spread across the AEC industry for an enhanced productivity.

In particular, it offers architects the possibility to intensify capabilities for new design features, to facilitate performance improvement for civil engineers, and even to forecast profitability so as to increase the overall market exposure. Manufacturing a building involves a meticulous and advanced spatial definition involved during each construction phase. Considering the building life-cycle in order to improve the building's structure or building services is simplified by BIM models.

BIM model specifications are developed by buildingSMART, a worldwide recognised authority: international standards are defined according to a set of five technical categories. First of all, carrying digital data is eased by *industry foundation classes* (IFC), recorded under the ISO 16739. The IFC schema lays upon a series of fundamental concepts. A detailed representation of a process can be found in its process map, described through the *information delivery manual* (idm). Translating descriptive processes into technical requirements is resolved by the use of *model view definitions* (mvd). A complete ontology description integrates into the logic of terms mapping covered within the *international framework for dictionaries* (ifd). A change coordination format is shown in the *BIM collaboration format* (bcf) to enhance communication and overall project quality.

In the AEC sector, practitioners measure characteristic levels of the physical representation through the level of content, also called Level Of Development, ranging from LOD100 to LOD500 (Table 1). The notion of Level of Development is progressively evolving through the notion of Level of Information Need (LOIN), a necessary basis required to develop notions of Level of Geometry (LOG) and Level of Information (LOI).

LOD 100	initial phase of concept
LOD 200	design and development
LOD 300	boundary representation, explicit geometric appearance, depth/height
LOD 400	closed to LOD4 CityGML
LOD 500	manufacturer design specification

Table 1: BIM Level Of Developments

Standards for integrating digitising of smart cities into global GeoSpatial initiatives are offered by the Open GeoSpatial Consortium (OGC). CityGML, is an officially developed data format concepted for exhaustively modelling a city. Although the current paper proposes a digital transfer from IFC to CityGML, the conversion from CityGML towards IFC is also explored by (Chognard *et al.*, 2018). A main advantage of CityGML resides in its capability to model a city in order to simulate diverse functions such as electricity distribution or consumption, traffic, noise and pollution propagation. In fact, the representation of city objects is made through instance objects accessible with deep ranges of data schemas, as exhaustively described in (Gröger *et al.*, 2012). For instance, creating a building instance can be made by the use of the building module, in *building.xsd*: the building model is central in CityGML as exposed by Kolbe *et al.* (2012) to demonstrate an application case of CityGML for disaster management. Similarly, CityGML offers five Levels Of Details (Table 2) to fully describe elements distributed at a neighbourhood scale, ranging from 0 to 4, four being the most concise representation of LODs.

A digital challenge to integrate IFC to CityGML consists in insuring the correct transfer of the original geometric content. CityGML carries the advantage to offer a simpler way to interpret geometry, concurrently IFC enables a deeper level of abstraction. Adding other considerations, IFC and CityGML classes characterise alternative vision types to integrate outdoor and indoor elements, thus forming the challenging logic of the semantic paradigm. The digital conversion from IFC to CityGML is nevertheless facilitated by a semantic overlap between Level Of Details and Level Of Developments. IFC aggregates predefined classes for 3D geometries, whereas CityGML requires the use of 2D surfaces to represent 3D elements.

Indeed, in order to define 3D geometries, CityGML (*i.e.* version 2.0 or 3.0) classes refer to the GML default schemas 3.2.1, so as to create 2D surface types such as abstract rings. CityGML is designed for city scales, as opposed to IFC: in fact a significant geometric bias could appear with the IFC format when physical objects are separated by important distances (Uggla and Horemuz, 2018) for small scales. Not considering a realistic ellipsoid but a flat earth assumption infers a geometric distortion up to 80 ppm for a longitudinal separation of 100 km, to be corrected with a scale factor to support the map projection.

Biljecki *et al.* (2016b) created city models by using dedicated random functions, based on rules applied to an initial set in order to generate buildings with random dimensions. The current paper's proposition consists in creating a CityGML dataset based on the original IFC model's geometry; however, an augmented version featuring extra shapes on the surfaces as developed in Stouffs and Krishnamurti (2019) is nevertheless a potential beneficial case for design purposes. The digital transfer from IFC to CityGML exposes a wide range of risks, namely due to the difference of paradigms within the two formats. As a consequence a grammar is used, inspired from the triple graph theory, in order to hedge against the digital risk occurring during the transfer. In fact, a scrupulous definition of rules from source to target data is an answer to control the model's data flow during the transition. A non-control of the digital risk generates a potential bias of the target instance, in terms of semantic or geometry. Furthermore, the proposal is based on reconciling those paradigms. Given the richness and the fundamental gap present between IFC and CityGML, the use of the triple graph theory is strongly advised as a first answer to decrease or eliminate the digital risk involved (Stouffs *et al.*, 2018). As an adapted tool, the triple graph theory enables to conserve the structure of the original data. The FZKViewer (Institute for Automation and Applied Informatics) allows to visualise the CityGML and IFC data formats.

LOD 0	building represented as footprint, useful for small scale simulations
LOD 1	building different storeys as parallelepipeds, in 3D
LOD 2	more accurate
LOD 3	additive information added, *i.e.* windows
LOD 4	interior representation

Table 2: CityGML Level Of Details

Considering the semantic involves to benefit from data schemas available in CityGML, plus to enrich specific data schemas. Cities and buildings answer different description requirements, and so are the CityGML/IFC modelling paradigms. The tested model is adapted for a LOD4 CityGML representation type, due to the considered design which implies the presence of windows, detailed roofs or wall curtains with specific frames and plates. In particular geometries such as advanced boundary representations for the roof physical description are present in the model. The paper's methodology proposes a strategy to fully operate a conservative digital transfer of the original LOD400-LOD500 representation towards a LOD4 in CityGML.

3 Background

There is no uniqueness in the methodology to convert IFC data sets through CityGML (Donkers *et al.*, 2016) when referring to the example of a two storey house aimed at being transferred in a Level Of Detail of type LOD3: neighbourhoods are represented by sets of 2D surfaces: meanwhile, a set of individual geometric strategies are selected to transform original IFC model's geometric elements.

The FME software (Feature Manipulation Engine), designed for data integration, is useful to deal with geometric objects (Kardinal Jusuf *et al.*, 2017) or generally to deal with IFC towards GIS data-based transformations. The method is powerful since it allows to concisely fetch the selected targeted IFC schema part of interest, to facilitate the generation of objects, and to fit the semantic enrichment.

Shell objects in IFC are considered (Floros *et al.*, 2017) and converted by the use of FME to extract outer shell so as to create CityGML LOD3 types. The geometric transfer relies on FME capabilities. The current paper's strategy goes a step further, by using the original abstraction definition without loss of abstraction. For instance, if a polynomial curve is defined according to a set of control points (as a cubic spline), the strategy exactly re-estimates the original cubic spline (a finite radius of curvatures between control points is an exactness criterion, as opposed to a linear interpolation for which the radius of curvature is infinite).

Donkers *et al.* (2016) extracted the geometric continuity in selecting solid type geometries from the IFC model so as to perform a set of dilation and erosion in order to obtain reshaped geometries. 3D solids are transformed into 2D surfaces associated with the creation of topological refinements. Tracking artefacts and fully conserving geometric elements are key to minimise the loss of precision when Computational Fluid Dynamics simulations are run, since an artificial additional drag force appears.

Biljecki *et al.* (2018) exposed occurring loss of accuracy in spatial analysis resulting from the use of various Level Of Details, or led by data sets with acquisition errors: a building positioning error provokes a bias on the wind flow distribution analysis. In fact, the degradation of the level of accuracy has a higher impact than the degradation of Level of Details.

Targeting an optimal geometric conversion is achieved by the use of a ray tracing strategy (Kang and Hong, 2018) in order to identify the external surface of a boundary representation (b-rep) which results from b-reps present in an IFC data format. A ray tracing is performed on the b-rep to extract external surfaces and create CityGML geometric instances. Multi-threading is particularly efficient to reduce the processing conversion time.

Given the size of original data files to convert, a multi-threading procedure is to be taken into consideration to optimise the processing time of the conversion. Extracting external surfaces relies on counting the number of rays emitted by a placed camera, derived from the full ray tracing algorithm EOR based on an even/odd rule to estimate whether a surface is external or not. The high computational requirement is tackled with the use of a Z-buffer test (Kang and Hong, 2018) to centralise the tested render surfaces as opposed to a test on all surfaces.

The strategy of the current paper is similar to the one in Deng *et al.* (2016) to generate 3D GIS geometries based on IFC geometric instances. First of all, the path location of the data in the model is similar to the current paper's model, in addition implicit geometry associated with the use of a dedicated transformation matrix is also present, however clipping geometries in the IFC schema are not considered. Arroyo-Ohori *et al.* (2018b) solved on the one hand the semantic in ifcOpenShell, and on the other hand the geometry in a C++ Computational Geometry Algorithms Library called CGAL, that has the capacity of measuring distances between the original model and the converted model in order to assess the efficiency of the conversion.

A dedicated strategic structure of LODs is employed (Arroyo-Ohori *et al.*, 2015) to convert the model towards the maximum Level Of Detail, and to select a range of specific strategies to lower the Level Of Detail by involving edges projections in order to implement the topological reduction (*i.e.* simplification) of the original 3D data structure: a chimney in LOD0 is therefore represented as a dot. The current paper aims at conserving the maximum Level Of Detail to fully conserve the geometry. Machine learning algorithms are used to estimate the semantic labelling (Rook *et al.*, 2016) of CityGML objects based on heuristic rules applied to geometric elements: terrains, roofs, walls or grounds are recognised through their intrinsic individual triangles orientations. Estimated labels are then validated against a stochastic metric, called kappa. A kappa ratio above 0.8 indicates a reliable automatic labelling (0.99 kappa for the Waldbruecke city model).

Achieving a data integration from IFC to CityGML answers a wider scope introduced with the concept of interoperability (Toth *et al.*, 2012). Data formats in general address specific types of description, outdoor enhanced detailing (CityGML) or indoor (IFC). Therefore, interoperability becomes a key issue to be addressed. The separation between those interpretations infers a data set description difference, solved by correctly addressing the interoperability. There is a need in integrating BIM standards (IFC) with generative evolutionary systems as exposed Janssen (2006) in the schema design method, with the seeding concept as a solution. The proposed method relies on rules to generate designs (evolutionary system), in addition to defining characters. Generative and evolutionary address design exploration (Stouffs and Rafiq, 2015) in addition to optimisation. A set of 11 related papers are presented in order to expose state-of-the-art systems, such as the Dexen platform. Those systems are designed to provide a set of optimal solutions, to be found along the Pareto boundary.

Although the current research developments allow to effectively transfer IFC datasets towards CityGML, the actual approach shows an alternative method to semantically enrich CityGML schemas, and it illustrates the efficiency of accessing the most detailed representation of the geometry in IFC datasets prior translating them in CityGML data format. The current methodology specifically requires the use of three representation graphs in order to perform a conversion. A first graph to describe the source IFC dataset, a linking graph to project nodes, and finally a target graph to handle both of the semantic and the geometry in CityGML. The graph specification highly depends on the nature of the model to convert. First of all, the source graph is specific to the model's structure, secondly the target graph allows to reach a predefined set of semantic structure, and finally the linking graph offers a unique way to join graphs between them.

4 General model description

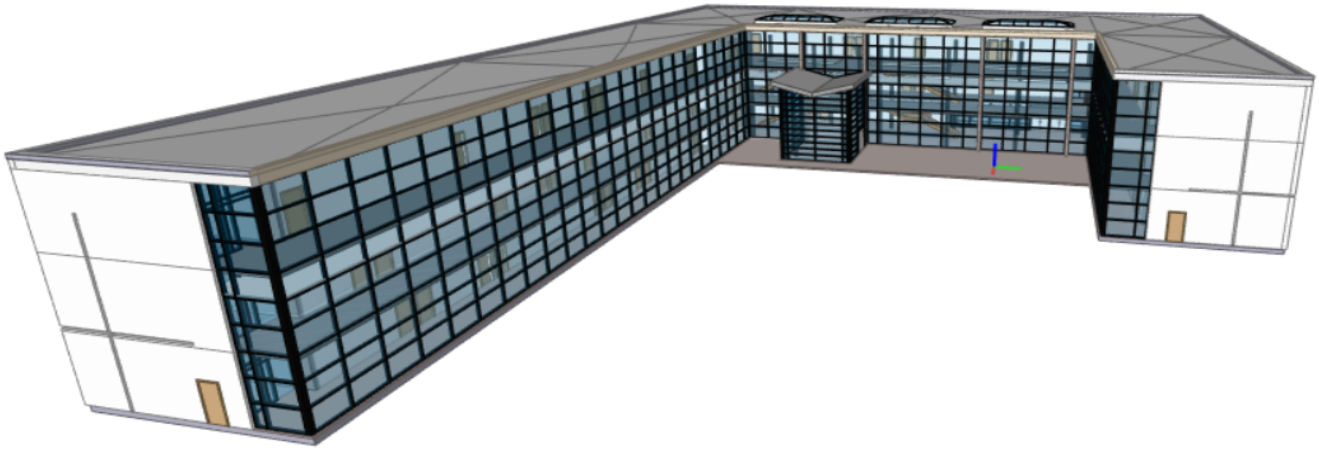


Figure 1: IFC selected model, global view

The selected model (Figure 1) scoped for an IFC to CityGML conversion is initially developed by Autodesk, and designed in Revit. The building in IFC format involves multiple storeys, where the model's structure is illustrated in the descriptive Table 4 in the case of *ifcBuildingElements* or *ifcSpatialElements*. The overall building model's geometry, semantic and attributes are stored in the project instance (*ifcProject*), from which the building instance (*ifcBuilding*) is initially extracted through an *ifcSite*.

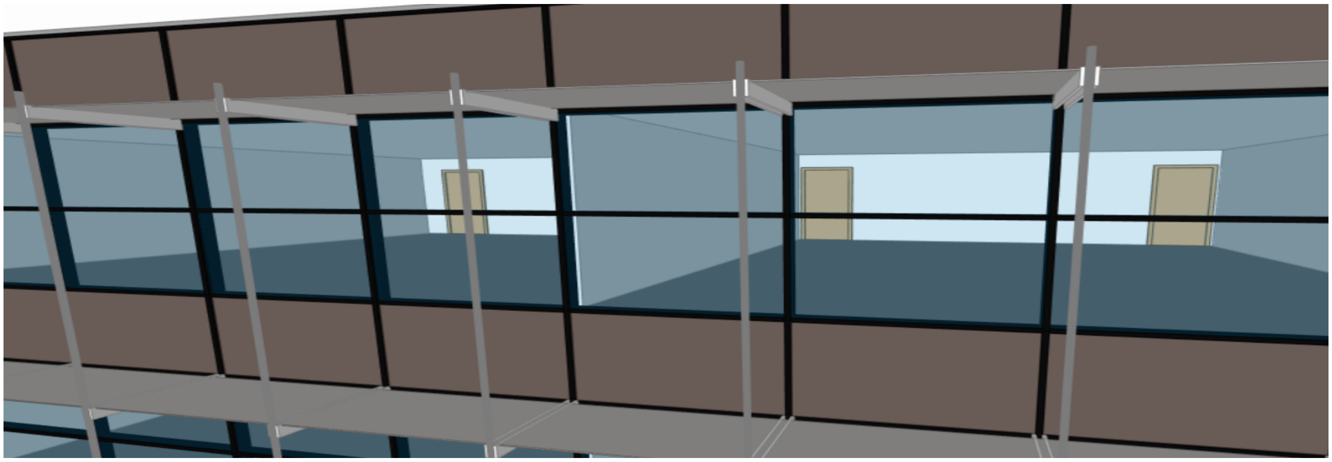


Figure 2: Outdoor frames

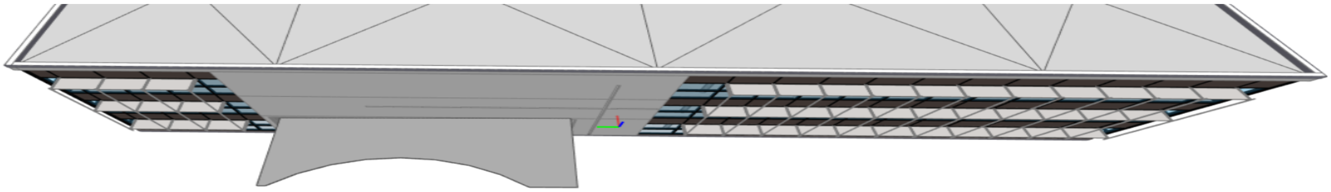


Figure 3: Elliptic back roof

Geometries of all objects are estimated in accordance with their *ifcProductRepresentation* and placed with their *ifcLocalPlacement*.

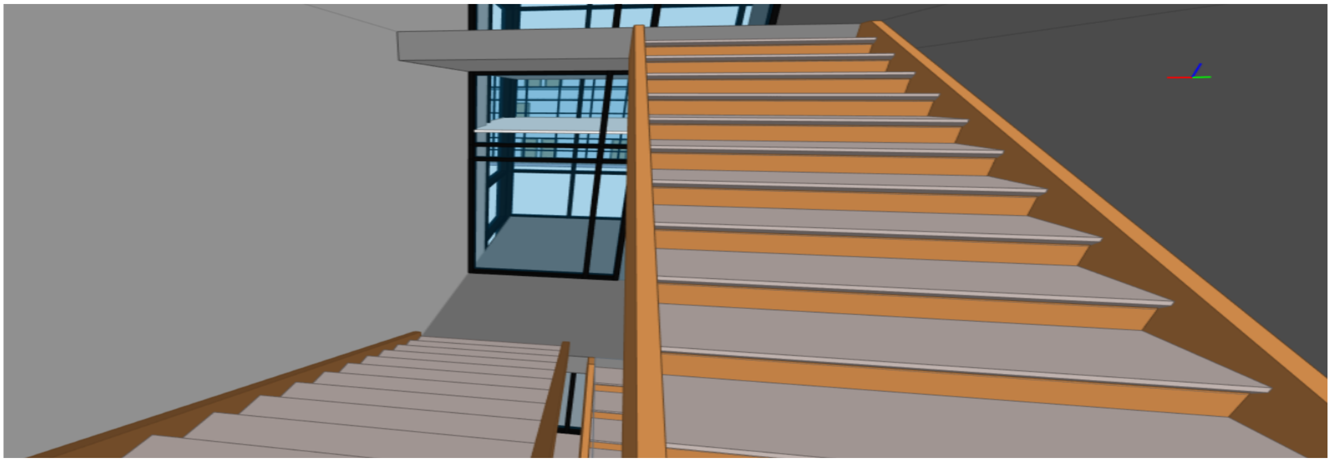


Figure 4: Indoor stairs

The model is a three storeys building composed with various sets of 3D objects, such as stairs, curved roofs, detailed curtain walls or windows (Figure 2). The model contains spatial elements (*ifcSpace* objects) defined as swept curves: individual curves are designed as arbitrary profile definitions with voids modelled as circles. In the proposed strategy, the full scope of building elements present in the model is converted, including various types of stairs (Figure 4), mapped doors with frames, and roofs.

An *ifcDoor* (Figure 5) geometry can be found through *ifcMappedItem* instances, given that the model contains a limited set of door types. Mapping a door requires the use of a rotation/translation matrix. In the model, doors are composed by a list of three individual items: a main part (swept and based on a rectangle extrusion), an outer frame (extruded from an arbitrary closed profile definition), and finally a third part to compose the inner frame.

The *ifcSpace* (Figure 6) instances are modelled as swept areas from arbitrary closed profile definition with voids. The closed profile is defined as a succession of points in a 2D plan, voids are defined as rectangles, circles, or polylines. Correctly placing an object in 3D requires its definition plus its referential placement expressed through a rotation and translation 4 by 4 matrix. If multiple referentials are considered, the equivalent matrix must be solved by recursively multiplying all matrices involved. The large scale considered in the model does not require a geodesic correction.

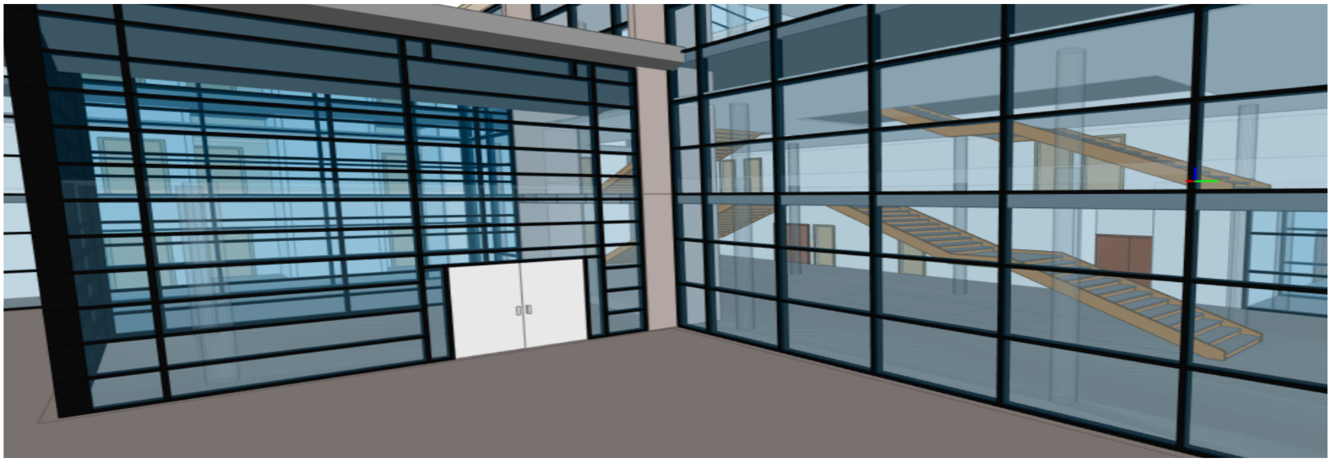
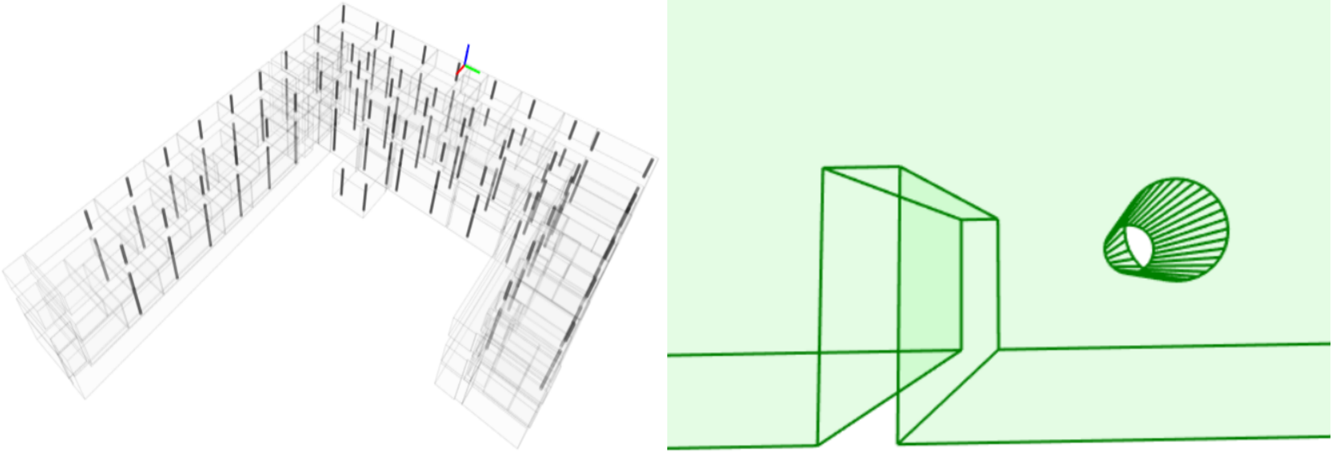


Figure 5: Double door for main entrance



(a) Global view of spaces in the selected IFC model

(b) Cylindrical void detail contained in an *ifcSpace*

Figure 6: Original spaces

5 Challenges of a conversion

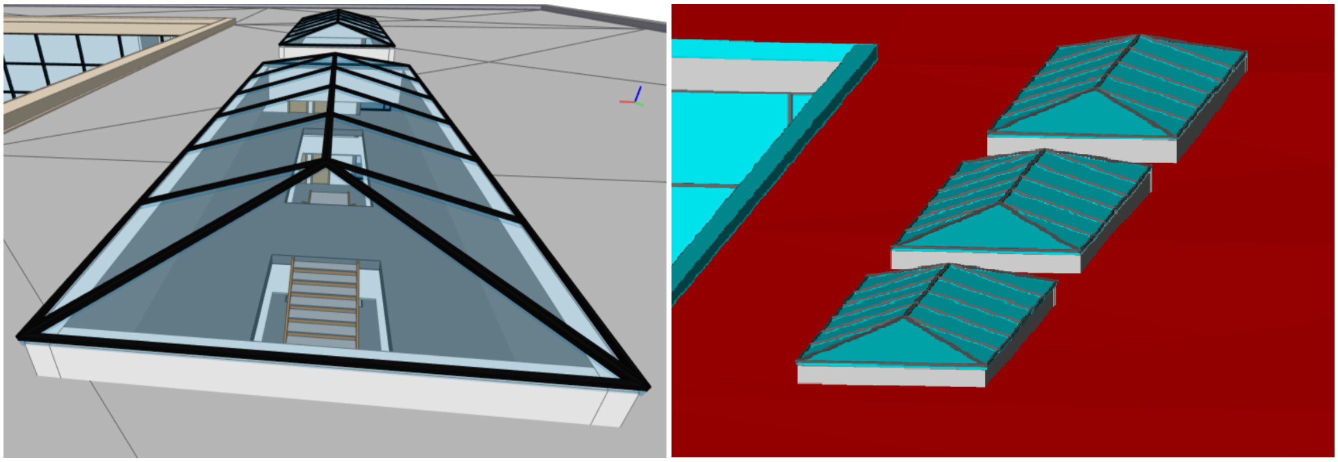
The digital transfer from IFC to CityGML carries a set of pre-identified challenges occurring during a conversion. This section aims at highlighting obstacles encountered and to propose solutions to avoid them. Geometry is a key element to be considered when a transfer occurs: the nature of associated challenges is commented (Table 3). As models of various sizes are taken into consideration, the execution time needs to be addressed. In addition, the quality control of the methodology through a set of unit testing framework is key.

Addressing the geometry: the proposed converting strategy consists in using geometric coordinates in the model to perform a loss less transformation. A voxel allows to solve complex geometries from IFC to CityGML integration (Arroyo-Ohori *et al.*, 2018a), however the scope of the paper's model counts limited types of geometries (Figure 7), simplifying thereupon the automation procedure.

Addressing the execution time optimisation: the conversion framework shows a repetition of identical operations. A multi-threading approach is priviledged in order to process object types simultaneously.

Addressing the methodology's quality: the Java-based implementation uses a set of unit tests in order to assess the quality of the functions deployed and the data created. Those unit tests are performed to control the consistency of the developed functions.

The Figure 7 shows a rooftop view before and after conversion. The Figure 7a is the original object selected, viewed as BIM Level of Development 400. The Figure 7b shows the same rooftop view after conversion in CityGML. A Level of Detail 4 is observed as well as satisfactory visual results.



(a) BIM Level Of Development 400 (b) CityGML Level Of Detail 4

Figure 7: Selected model's rooftop

Challenge		Solution
semantic	⇒	The model tested shows a category of semantic which is not present in the CityGML schema. A dedicated mapping in the context of a domain extension is proposed.
geometric	⇒	The tested model shows various types of challenges, such as edges non listed by order, the methodology proposes to adapt the geometric conversion to address this shortcoming.

Table 3: Answering shortcomings intrinsic to the data transfer

6 Methodology

A successful data conversion strategy consists in the use of an approach inspired from the graph theory (Stouffs *et al.*, 2018). The triple grammar graph, composed with computational and mathematics concepts, is applicable in the case of the current digital transfer from IFC to CityGML, and more generally for alternative data schemas type used in the architecture domain. Three main abstract categories are considered: the triple graph is a combination of the source graph (IFC), the targeted graph (CityGML), and the linking one (IFC to CityGML). The triple graph grammar eases to overpass both of the semantic and the geometric risk during the digital transfer, in particular by enhancing the visibility of the data flow considered.

In order to conserve a consistent digital transfer, the geometry and the semantic are separately solved in the approach. The processing flow chart shows a detailed representation of how steps are followed during the digital transfer. The below methodology is proposed:

1. **Step 1/4 of process**
    (a) scoped object: IFC data source instance
    (b) data random check
    (c) visual detailed check
    (d) model structure identification

2. **Step 2/4 of process**
    (a) scoped object: IFC data source instance
    (b) scanning the model according to the predifined structure
    (c) infering and creating geometrical shapes based on the IFC geometric specifications according to the IFC schema

---

3. **Step 3/4 of process**
    (a) data exchange: from IFC to CityGML
    (b) mutual appropriate reconciliation of IFC to CityGML instances, according to a predifined CityGML schema domain extension

---

4. **Step 4/4 of process**
    (a) scoped object: CityGML data target instance
    (b) data random check
    (c) visual detailed check
    (d) model structure control

As a first step, the nature of the model is investigated through preliminary visual checks, plus random checks in the IFC model so as to get general and relevant details (schema of type IFC4 for instance) about the data to convert, then, the overall structure content of the data model is checked. Further checks can be additionally computed through the use of interoperability tools or softwares like Solibri.

Tables 4 and 5 illustrate the model structure identification and the geometric map. In the exposed case, *ifcBuilding* is composed by *ifcStorey* instance objects to gather spatial or structural elements such as *ifcSpace* or *ifcWall* or *ifcCurtainWall* or *ifcWindow* or *ifcDoor* object instances (Table 4). Those elements are then represented by a collection of items with a geometric representation. The *ifcWindow* is modelled as a rectangle extrusion and present in the model through an *ifcMappedItem* (Table 5) or an *ifcExtrudedAreaSolid.* The *ifcWall* contains opening options in order to allow the presence of windows or doors. The *ifcCurtainWall* is decomposed by a series of *ifcBuildingElement* instances which include *ifcMembers* or *ifcPlates.* Similarly to *ifcWall* instances, the *ifcSlab* contains voids, in order to enable space communication from a level to another and to allow stairs to go through. The *ifcStair* is modelled as a collection of surfaces individually composed by extruded closed lines (*ifcArbitraryClosedProfileDef*). The *ifcRoof* is geometrically modelled by an instance of *ifcAdvancedBrep* object, defined through a set of oriented vertices. In the model, elliptical objects are present (instance of *ifcEllipse*, Figure 3).

A second step consists in a deeper and concise analysis of the data structure in order to identify the location of the geometry in the data model, resulting in the identified semantic path to follow. Once, the location of the geometry is specified, the content of the geometry is analysed in order to accordingly transfer abstract mathematical objects found. For instance, a rectangle is a 2D abstraction defined through the use of three parameters: an origin point, and two lengths. As a third step, the data exchange from IFC to CityGML occurs when appropriate target instances are found (in the logic of the Table 7): bldg:Building or bldg:BuildingPart are examples of target instances in CityGML. The fourth step consists in validating the transferred data. The result is analysed from a visual point of view in addition to random checks in the CityGML generated data model, coupled with visual controls of the semantic through colour checks. A validation table is built so as to identify and count initially transferred objects from IFC compared with those converted in CityGML instances (Table 11).

The methodology exposed in the current paper is beneficial to address both of the geometry and semantic during a digital transfer from IFC to CityGML. A correct visualisation relies on an efficient geometric transfer, in addition, keeping the differentiation type between objects is attractive for architects/civil engineers, and can be made in accordance with the semantic transfer. Given that the geometric transfer relies on specific geometries encountered in the model, it seems that a preliminary geometric inventory is required in order to state the different forms occurring in the model. The principle being the following: individual elementary components in the geometries are fetched, then re-built through the use of boundary representations.

ifcBuilding ⇒	**ifcCovering** ⇒	Representation ⇒	**ifcProductRepresentation**		
	↘	ObjectPlacement ⇒	**ifcLocalPlacement**		
	↘ **ifcWindow**	…			
	↘ **ifcMember**	…			
	↘ **ifcPlate**	…			
	↘ **ifcSlab**	…			
	↘ **ifcRoof**	…			
	↘ **ifcWall**	…			
	↘ **ifcWallStandardCase**	…			
	↘ **ifcStairFlight**	…			
	↘ **ifcBuildingElementProxy**	…			
	↘ **ifcDoor**	…			
	↘ **ifcCurtainWall** ⇒	IsDecomposedBy			
		⇓			
		ifcRelAggregates			
		⇓			
		RelatedObjects ⇒	**ifcPlate** ⇒	Representation ⇒	**ifcProductRepresentation**
			↘	ObjectPlacement ⇒	**ifcLocalPlacement**
			↘ **ifcMember**	…	
	↘ **ifcStair** ⇒	IsDecomposedBy			
		⇓			
		ifcRelAggregates			
		⇓			
		RelatedObjects ⇒	**ifcStairFlight** ⇒	Representation ⇒	**ifcProductRepresentation**
			↘	ObjectPlacement ⇒	**ifcLocalPlacement**
			↘ **ifcSlab**	…	
			↘ **ifcMember**	…	
ifcSpatialElement ⇒	**ifcSpace** ⇒	Representation ⇒	**ifcProductRepresentation**		
	↘	ObjectPlacement ⇒	**ifcLocalPlacement**		

Table 4: Building model structure description

7 Implementation

The logic of data standards enrichment is inscribed in the development of CityGML throughout the release of the following multiple versions (1.0, 2.0 or 3.0). Features present in the standards allow a more accurate and concise representation of cities by accentuating the level of description of individual elements. For instance, CityGML 3.0 proposes abstract elements called AbstractOccupiedSpace (in the CityGMLBase module) to differentiate implicit representations for LOD1, LOD2 or LOD3. This improvement, compared with CityGML 2.0 contributes to facilitate the reduction of the semantic gap between IFC and CityGML. In the current proposition, a double action is made on existing CityGML schemas, in order to ease the semantic solving.

1. A primary modification of the building module (*building.xsd*) schema to allow new types of elements in the scene, *i.e. bldg:Door*, or *bldg:Window*. Those are of type *BuildingPartType* and to be substituted in the *_AbstractBuilding* substitution group.
2. A modification of the cityGMLBase module (*cityGMLBase.xsd*), in order to allow the association of the above created building elements. The methodology, therefore, proposes to enrich the complex of type *AggregatesPropertyType* by extending its complex content based from *gml:AssociationType* with all elements mentioned above, *i.e. bldg:Door* or *bldg:Window*.

		ifcRepresentationContext		
		⇓		
HasSubContexts	⇐	ifcGeometricRepresentationContext		
⇓				
ifcGeometricRepresentationSubContext	⇒	ContextIdentifier	*criteria 1* ⇀	Body
	⇒	ContextType	*criteria 2* ⇀	Model
	⇒	RepresentationsInContext	*criteria 3* ⇀	ifcShapeRepresentation
				⇓
				Items or ifcMappedItem
				⇓
		ifcExtrudedAreaSolid	ifcBooleanClippingResult	ifcAdvancedBrep
		⇓	⇓	⇓
		ifcProfileDef	**ifcBooleanOperand**	**ifcTopologicalRepresentationItem**
		ifcArbitraryClosedProfileDef		*ifcClosedShell*
		ifcRectangleProfileDef		
		ifcCircleProfileDef		
		ifcArbitraryProfileDefWithVoids		

Table 5: Model geometric map

The approach is encouraged since there is a consequent advantage in its singularity. In fact, more precision and contact with original coordinates improve the authenticity of the geometric transfer from IFC to CityGML. The methodology requires a pre-identification of geometric types present in the model in order to correctly perform boolean operations of CSGs.

The approach integrates with three graphs: Table 6 shows an extraction of the source graph, Table 7 describes the linking graph, and finally the third graph involved represents the targeted graph in CityGML.

7.1 semantic solving

The model contains elements with voids, such as *ifcSpace*, therefore mapping them with pairs of gml:exterior and gml:interior linear rings are preferred.

ifcBuilding ⇒	IsDecomposedBy					
	⇓					
	ifcRelAggregates					
	⇓					
	RelatedObjects ⇒	**ifcBuildingStorey** (1) ⇒	ContainsElements			
			⇓			
			ifcRelContainedInSpatialStructure			
			⇓			
			RelatingStructure ⇒	**ifcSpatialElement** ⇒	Representation ⇒	**ifcObjectPlacement**
				↘	ObjectPlacement ⇒	**ifcLocalPlacement**
		(2) ⇒	ifcDecomposedBy			
			⇓			
			ifcRelAggregates			
			⇓			
			RelatedObjects ⇒	**ifcBuildingElement** ⇒	Representation ⇒	**ifcObjectPlacement**
				↘	ObjectPlacement ⇒	**ifcLocalPlacement**
	↘	**ifcBuildingStorey**	…			
		⋮				
	↘	**ifcBuildingStorey**	…			

Table 6: Semantic path observed in selected building model

Once the structure of the path to access the geometry is identified (Tables 5 and 6), the methodology refers to Table 7 so as to match and map IFC instances towards CityGML instances. For example, an *ifcBuilding* instance is mapped with a bldg:Building instance, and an *ifcSpace* with a bldg:Room.

ifcBuilding	⇒	bldg:Building	ifcWindow	⇒	bldg:Window
ifcBuildingStorey	⇒	bldg:BuildingPart	ifcCovering	⇒	bldg:FloorSurface
ifcSpace	⇒	bldg:Room	ifcCurtainWall	⇒	bldg:WallSurface
ifcWallStandardCase	⇒	bldg:WallSurface	ifcMember	⇒	bldg:FloorSurface
ifcWall	⇒	bldg:WallSurface	ifcPlate	⇒	bldg:BuildingPart
ifcSlab	⇒	bldg:FloorSurface	ifcBuildingElementProxy	⇒	bldg:BuildingPart
ifcRoof	⇒	bldg:RoofSurface	ifcStair	⇒	bldg:BuildingPart
ifcDoor	⇒	bldg:Door	ifcStairFlight	⇒	bldg:BuildingPart

Table 7: Mutual instance correspondences

7.2 semantic enrichment in CityGML scheme

Inconsistencies may happen during data transfers, such as geometries (Biljecki *et al.*, 2016a). In addition, it appears that a descriptive ontology to define the structure of physical representations may occur (Stouffs and Tuncer, 2015). The current paper proposes to solve the underlying semantic gap by enriching fundamental CityGML schemas. Enriching the CityGML schema enables to develop a more precise representation of semantic relations between building element instances, such as the link bldg:Building to bldg:BuildingPart. This relation is modified in order to name it accordingly with the IFC data structure: the link is then called *consistsOfAggregates*. In order to achieve the link, the following modifications are proposed:

1. extending the complex of type *AggregatesPropertyType* by adding *bldg:Building* and *bldg:BuildingPart* elements.
2. since *bldg:Building* and *bldg:BuildingPart* inherit from the abstract complex of type *AbstractSiteType*, a modification consists in extending its complex type by adding the element *consistsOfAggregates* of type *AggregatesPropertyType*.

The use of the triple graph is encouraged to avoid semantic mapping inconsistencies, *i.e.* mapping a roof instance instead of a wall for instance, and to preserve or enrich the nature of the links between CityGML instances.

7.3 geometric solving

The geometric objects present in the scene also need a semantic solving: geometries are mapped with instances of *lod4MultiSurface*. The model contains geometric features such as *ifcArbitraryClosedProfileDef* (Table 8), mainly composed by *ifcPolylines*, the rest being sets of *ifcCompositeCurve* or *ifcTrimmedCurves*.

Table 9 is a practical case to illustrate how the geometric *ifcClosedShell* is solved.

ifcPolyline	335
ifcCompositeCurve	9
ifcTrimmedCurve	8

(a) Exploring *ifcArbitraryClosedProfileDef*

(inner and outer)	
ifcCompositeCurve	156
ifcPolyline	66

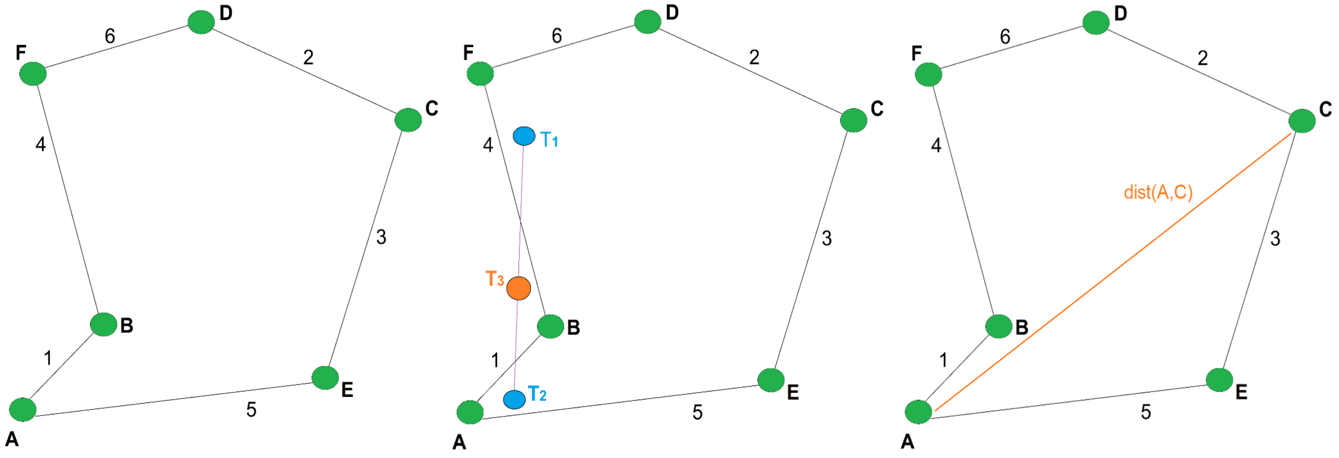
(b) Exploring *ifcArbitraryProfileDefWithVoids*

Table 8: Exploring geometric entities in the selected model

In case an *ifcClosedShell* encountered in the model defines a concave closed loop as shown in Figure 8c, a dedicated function is required to efficiently rebuild the closed polyline (Table 10).

ifcClosedShell
⇓
CfsFaces
⇓
ifcAdvancedFace
⇓
Bounds
⇓
ifcFaceOuterBound
⇓
Bound
⇓
ifcEdgeLoop
⇓
EdgeList
⇓
ifcOrientedEdge ⇒ Establishing Connection *set of edges in random order*
⇒ solving the connection through a distance function

Table 9: Proposition to geometrically solve the *ifcClosedShell* entity



(a) Original 2D closed loop (b) Geometric concavity (c) Solving the concave closed loop

Figure 8: Solving random 2D closed loop present in model

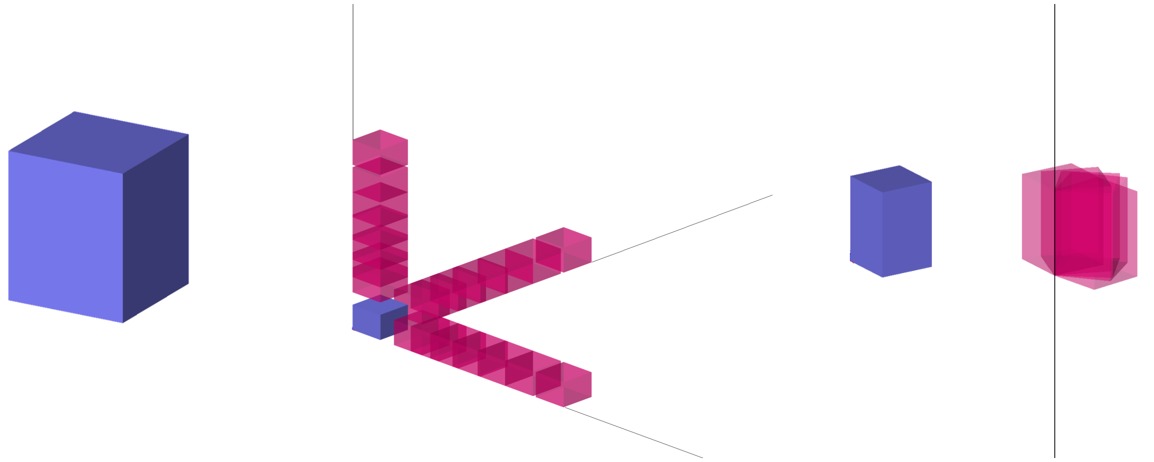
input a set of segments E, and a segment XY or a polyline
output the selected segment
⇀ *proposed algorithm design*

```
findNextSegment(E, XY){
    E' = E.copy(); E'.delete(XY)
    min = +infinity; segmentChoice = null
    for(Segment seg in E'){
        if(dist(Y, seg(a)) <min){
            segmentChoice = seg; min = dist(Y, seg.getFirstPoint())
        }
        if(dist(Y, seg(b)) <min){
            segmentChoice = seg;
            min = dist(Y, seg.getSecondPoint())
        }
        if(min <threshold){return segmentChoice}
    }
}
```

polyline	closed	.findNextSegment()	.formPolyline()
AB	FALSE	BF	ABF
ABF	FALSE	FD	ABFD
ABFD	FALSE	DC	ABFDC
ABFDC	FALSE	CE	ABFDCE
ABFDCE	FALSE	EA	ABFDCEA
ABFDCEA	TRUE		
		↘	
			return ABFDCE

Table 10: Algo, findClosedPolyline()

The *ifcLocalPlacement* is central in the geometric solving: it places an object in 3D and involves multiple referentials, series of rotation/translation matrices are therefore involved in order to achieve the correct placement in 3D (Figure 9).



(a) Initial 3D object

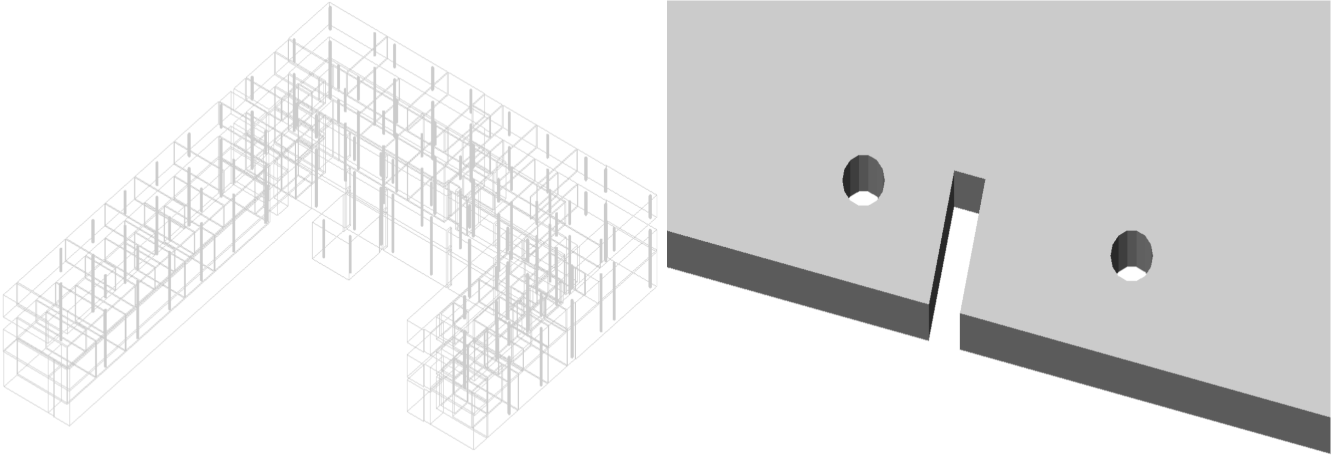
(b) Series of translations, three axis

(c) Series of rotations, single axis

Figure 9: 3D rotation and translation

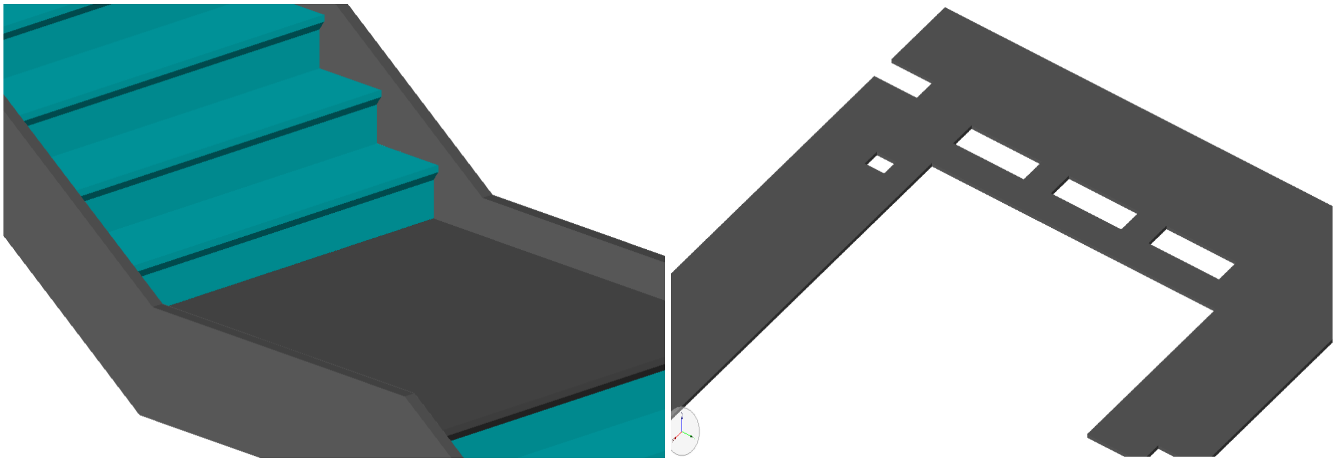
8 CityGML modelled target instance

The resulting space/Room modelled (Figure 10a and Figure 10b), shows the expected targeted semantic, *i.e.* grey coloured, and the expected targeted geometry with voids, performed with a series of classical CSGs boolean operations. The exposed converted *ifcBuildingElement* involves objects as *ifcDoors* (Figure 11b), *ifcRoofs* (Figure 11a), *ifcCurtainWalls* (Figure 11c), *ifcStairs* (Figure 10c), *ifcSlabs* (Figure 10d) and their voids. The detailed door representation is correctly modelled: the double frame door and instances of *ifcMappedItems* are well transferred. Mapped items require the use of a 3D rotation/translation matrix in order to be solved. The user predefined semantic colour of a door, blue, is in line with the one on the picture.



(a) Global view of bldg:Room entities

(b) Cylindrical voids present in geometries



(c) Details of indoor stair

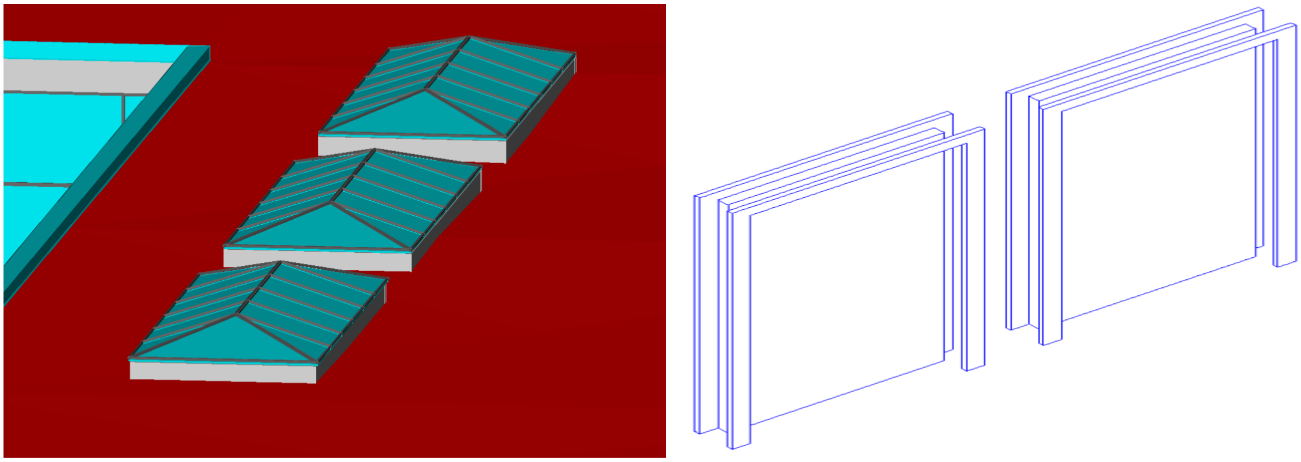
(d) Slab with voids

Figure 10: CityGML target instances resulting from the selected model's conversion

As expected, the slab representation of the storey in the building correctly involves the presence of voids. Those voids are formed by swept rectangles, which belong to an inner linear ring of a surface member to form an instance of the CityGML slab which colour is grey. The side model representation of frames for the outdoor curtain wall is also correctly represented. A crossed view (Figure 11d) of the targeted building is made in order to check the consistency of the transfer.

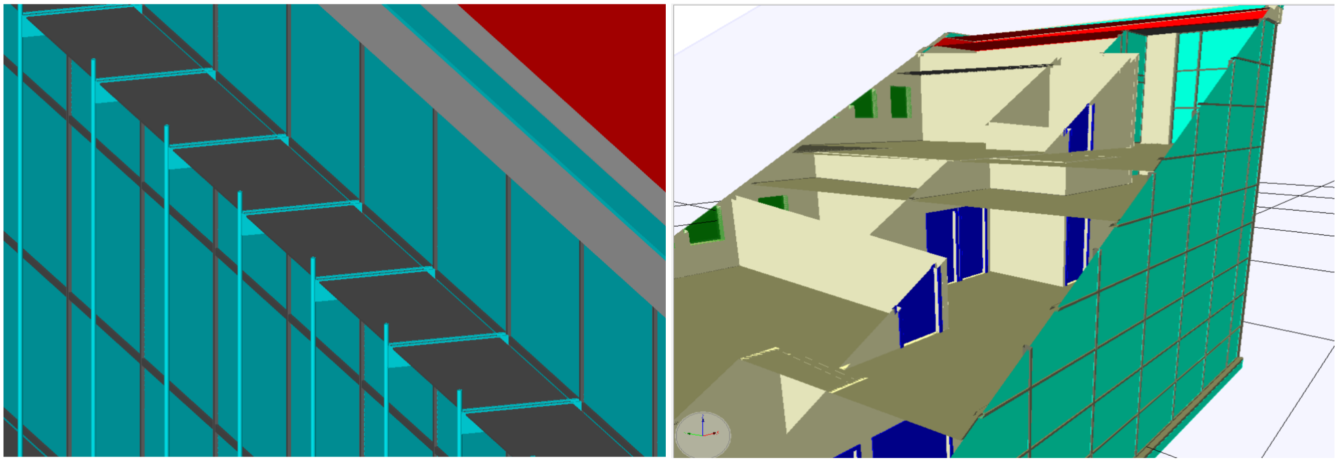
CSGs are present in the model, given that boolean clipping operators are used. These geometric (Figures 10b and 10d) considerations require the use of routines, developed in sets of algorithms. Those are necessary to correctly perform boolean operations between walls and voids or openings. A CSG is defined as a boolean substraction, and can be considered as a special case derived from the sortal approach (Stouffs, 2008) although more complex boolean operators based on shape algebras exist (Stouffs, 2018).

The tested model shows a Level Of Development of type 400, according to the collection of objects present in the scene.



(a) rooftop view

(b) Doors and frames



(c) Outdoor frames and curtain walls

(d) Outdoor/Indoor cross section

Figure 11: Various set of CityGML instances resulting from the selected model's conversion

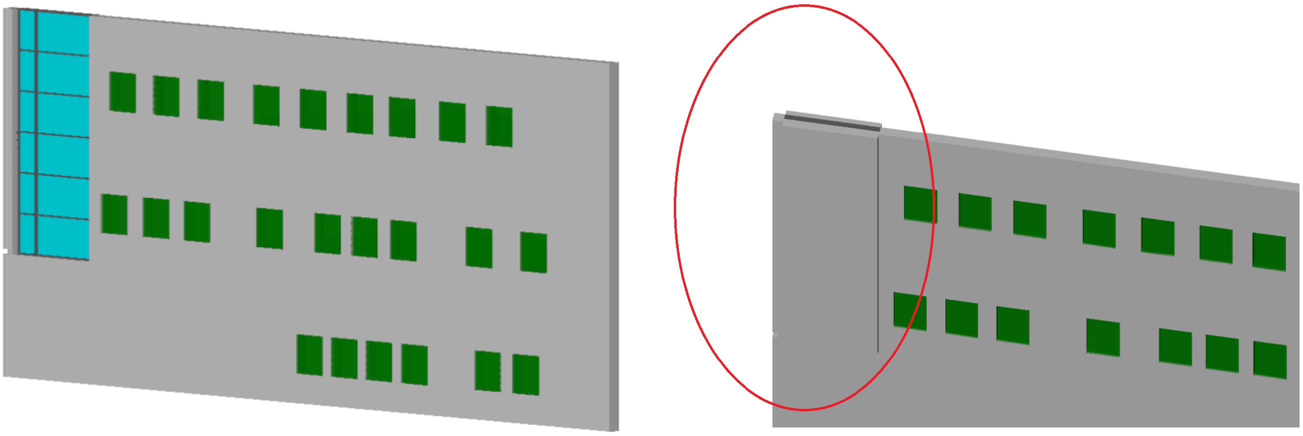
9 Discussion: specific modelling challenges

The conversion in itself is not primarily straightforward in the sense that artefacts were initially found during transfer. Those needed to include additive routines in order to be corrected.

The converting strategy uses CSGs to reconstitute forms resulting from boolean operations. As shown above, some cases for which a correction is necessary were encountered. In particular, in the case openings are not strictly parallel to the facade it seems that the strategy considers the opening as a non-void, which is geometrically false (Figure 12). Hence in order to correct that detail, an additive routine to reshape the openings is included. The routine consists in re-estimating the opening proportions by chopping it down to the basis of the concrete. That reshape prevents a set of curtain wall from being obstructed by an artefact.

Moreover, this case appeared in the south facade, which is composed of a wall with an empty double cross in its centre. This empty double cross is modelled as an opening and belongs to the same category of correction: in order to overpass it, a chopped version had to be taken into consideration in order to obtain a facade which looks like the original one. In fact, those details are important to correct for the reason that a use of that CityGML model in a Computational Fluid Dynamics Analysis leads to a wrong wind flow velocity estimation.

The presence of a cavity for a viscous flow generates a vortex which is supposedly non-present when the facade is modelled by the use of the initial converting strategy. A Computational Fluid Dynamics simulation performed under the Lattice Boltzmann Method was used in order to illustrate that difference (Figure 13).



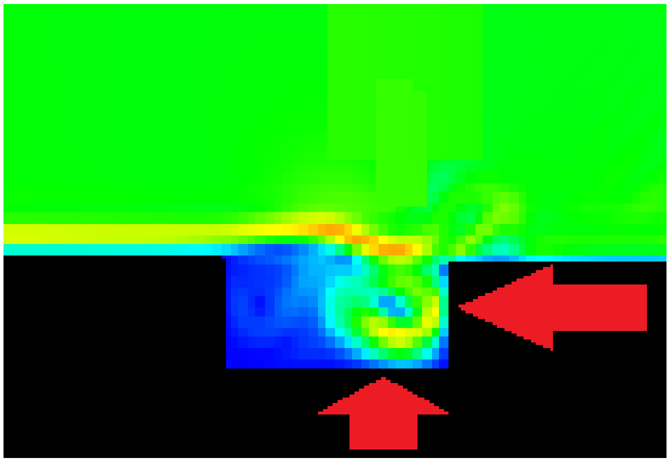
(a) Expected facade conversion (b) Unexpected facade obstruction

Figure 12: Correcting unexpected obstructions

More generally, an accurate geometric representation reduces the bias in estimating the roughness coefficient for outdoor environmental simulations. Those types of widened openings are largely present in the model, correcting them is a requirement.

Concurrently to the principle of creating extra *xsd* schemas through an Application Domain Extension, and in order to keep an equivalent objective so as to facilitate the conservation of predefined link types, the current paper's methodology opts for a semantic strategy consisting in modifying existing *xsd* files available in the collection of CityGML schemas (*cityGMLBase* and *building* in the case of CityGML 2.0). Considering CityGML 3.0 would imply to modify *cityGMLBase*, *building* and *construction* modules.

`01bDresultFacadeartefact.png`



(a) Obstructed cavity (b) Expected vortex in cavity

Figure 13: Computational Fluid Dynamics analysis for geometric impact estimation

10 Validation

The first criterion retained to validate the proposed results consists in proceeding with a visual check of the CityGML instance generated.

The second criterion consists in performing the counting of instances in the IFC model compared with those achieved in the CityGML model. The Δ is the counting difference obtained. A Δ equal to zero means that the number of objects initially present in the original model is correctly mapped.

A third criterion is defined by the semantic which is then reflected in the colours of the CityGML instances generated, *i.e.* a blue door, a red roof, a grey slab.

	IFC			*total*	Δ	*total*	**CityGML**
	ifcBuilding						bldg:Building
	1			1	0	1	1
	ifcSpace						bldg:Room
	91			91	0	91	91
	ifcDoor						bldg:Door
	100			100	0	100	100
	ifcWindow						bldg:Window
	24			24	0	24	24
ifcSlab	ifcCovering	ifcMember					bldg:FloorSurface
29	65	3308		3402	0	3402	3402
	ifcRoof						bldg:RoofSurface
	1			1	0	1	1
ifcWallStandardCase	ifcWall						bldg:WallSurface
166	6			172	0	172	172
ifcStairFlight	ifcPlate	ifcBuildingElementProxy	ifcBuildingStorey				bldg:BuildingPart
20	1349	50	5	1424	0	1424	1424

Table 11: Validation in model 1

IFC	*total*	Δ	*total*	**CityGML**
ifcBuilding				bldg:Building
1	1	0	1	1
ifcSpace				bldg:Room
6	6	0	6	6
ifcWall				bldg:WallSurface
12	12	0	12	12
ifcSlab				bldg:FloorSurface
2	2	0	2	2
ifcRoof				bldg:RoofSurface
1	1	0	1	1
ifcBuildingStorey				bldg:BuildingPart
3	3	0	3	3

Table 12: Validation in model 2

The model exposed in the current article is correctly validated, as shown in Table 11 and called model 1. In order to diversify the validation with different types of models, two extra models were considered, by introducing some variability, a first one, model 2, significantly smaller than model 1, and a second one, model 3, significantly bigger. Table 12 shows a correct validation since Δ is systematically equal to zero, and similarly Table 13 shows a correct validation, for the same reason.

Those results are globally considered as valid given an overall respected semantic plus a fairly detailed building elements and spaces representation. The consolidated Δ result found, zero, encourages and validates the results.

	IFC			*total*	Δ	*total*	**CityGML**
	ifcBuilding						bldg:Building
	1			1	0	1	1
	ifcSpace						bldg:Room
	722			722	0	722	722
	ifcDoor						bldg:Door
	756			756	0	756	756
	ifcWindow						bldg:Window
	125			125	0	125	125
ifcSlab	ifcCovering	ifcMember					bldg:FloorSurface
1380	92	3997		5469	0	5469	5469
	ifcWall						bldg:WallSurface
	6867			6867	0	6867	6867
ifcStair	ifcStairFlight	ifcPlate	ifcRampFlight				bldg:BuildingPart
20	79	1267	22				
ifcRamp	ifcBuildingElementProxy	ifcBuildingStorey					
6	1907	16		3317	0	3317	3317

Table 13: Validation in model 3

11 Conclusion

The scope of the proposed strategy answers shortcomings present during the conversion from IFC towards CityGML. A consistent digital transfer from IFC to CityGML relies on a set of two main issues, namely solving the semantic and solving the geometry. In terms of geometry, finding the correct path of the geometry is key, in addition to solve particular cases. For instance, the example of solving an *ifcEdgeLoop* instance present in the model is exposed by considering a dedicated algorithm. Positive advantages of the paper's proposition are also presented by testing the methodology against concave surfaces potentially present in BIM-IFC data sets. In the paper's model, an *ifcEdgeLoop* is defined as a list of *ifcOrientedEdges* instances in a non expected order; therefore the original closed edgeLoop must be recomputed before mapping it towards CityGML geometric instances. The developed methodology proposes to solve both of the geometry and the semantic during the conversion. In particular, a set of additive algorithms are developed to solve boolean clippings since they occur in wall or slab openings. Overall, the proposed methodology is validated against the number of counted instances present in CityGML compared with those existing in the original IFC model. Addressing the geometric integration in an IFC to CityGML conversion relies on the nature of the specific geometric instances found in the model in addition to the specific location of the geometry in the model: in particular, the tested model shows a root node called *ifcProject*. The triple graph grammar methodology (Stouffs *et al.*, 2018) is a key framework to control the data flow during the digital transition, therefore advantageous to face IFC to CityGML interoperability. Maximising the use of schemas available in CityGML is to be considered when BIM-IFC sources are involved in order to reinforce the mapping performances. Reproducing the exposed strategy requires the use of an IFC model, in addition to a set of predefined modifications of CityGML schemas. Once the path of the data is correctly identified, and the geometry consistently reconstituted in boundary representations in a CityGML LOD4, the results may be visualised in an FZKViewer. The main advantage of that method is to control the data flow, especially for geometric processing, which is advantageous to prevent from potential artefacts. The method shows positive aspects in terms of parallelization. A limitation of the presented method resides in the way the geometry is explicitly stored in the model, which consequence is to artifically increase the size of the generated CityGML target instance.

Acknowledgements

This material is based on research/work supported by the National Research Foundation under Virtual Singapore Award No. NRF2015VSG-AA3DCM001-008. The authors would like to acknowledge the NUS Research team (Architecture Department) and the OS Team (Ordnance Survey Great Britain): Helga Tauscher (NUS), Salman Khalili Araghi (NUS), Amol Janardan Konde (NUS), Lim Yan Yee Joie (NUS), Jeremy Morley (OS), Carsten Roensdorf (OS), James Crawford (OS), Diana Moraru (OS), Simon Lawrence (OS).

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Arroyo-Ohori, K., *et al.*, 2018a. Processing BIM and GIS models in practice: experiences and recommendations from a geoBIM project in the Netherlands. *ISPRS International Journal of Geo-Inf*, 7, 311.

Arroyo-Ohori, K., *et al.*, 2015. Modeling a 3D City Model and Its Levels of Detail as a True 4D Model. *ISPRS International Journal of Geo-Information*, 4 (3), 1055 – 1075.

Arroyo-Ohori, K., *et al.*, 2018b. Processing BIM and GIS Models in Practice: Experiences and Recommendations from a GeoBIM Project in The Netherlands. *ISPRS International Journal of Geo-Information*, 7 (8), 311.

Biljecki, F., *et al.*, 2018. The effect of acquisition error and level of detail on the accuracy of spatial analyses. *Cartography and Geographic Information Science*, 45 (2), 156–176.

Biljecki, F., *et al.*, 2016a. The most common geometric and semantic errors in CityGML datasets. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W1, 13–22.

Biljecki, F., Ledoux, H., and Stoter, J., 2016b. Generation of multi-LOD 3D City models in CityGML with the procedural modelling engine random3DCity. *Remote Sensing and Spatial Information Sciences*, IV-4/W1, 51–59.

Chognard, S., *et al.*, 2018. Digital construction permit: a round trip between GIS and IFC. *Advanced Computing Strategies for Engineering, 25th EG-ICE International Workshop.*

Deng, Y., Cheng, J.C., and Anumba, C., 2016. Mapping between BIM and 3D GIS in different levels of detail using schema mediation and instance comparison. *Automation in Construction*, 67, 1–21.

Donkers, S., *et al.*, 2016. Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings. *Transactions in GIS*, 20 (4), 547 – 569.

Floros, G., Pispidikis, I., and Dimopoulou, E., 2017. Investigating integration capabilities between Ifc and CityGML lod3 for City Modelling. *Energy Procedia*, XLII-4/W7, 1–6.

Gröger, G., *et al.*, 2012. OGC City Geography Markup Language (CityGML) En-coding Standard. *Open GeoSpatial Consortium*, 1–326.

Janssen, P., 2006. A generative evolutionary design method. *Digital Creativity*, 17, 49–63.

Kang, T.W. and Hong, C.H., 2018. IFC-CityGML LOD Mapping Automation Using Multiprocessing-based Screen-Buffer Scanning Including Mapping Rule. *KSCE Journal of Civil Engineering*, 22 (2), 373–383.

Kardinal Jusuf, S., *et al.*, 2017. Integrated modeling of CityGML and IFC for city/neighborhood development for urban microclimates analysis. *Energy Procedia*, 122, 145–150.

Kolbe, T.H., Gröger, G., and Plümer, L., 2012. CityGML - Interoperable Access to 3D City Models. *Institute for Cartography and Geoinformation*, 1–16.

Rook, M., Biljecki, F., and Diakité, A., 2016. Towards automatic semantic labelling of 3D City Models. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W1, 23–30.

Stouffs, R., 2008. Constructing design representations using a sortal approach. *Advanced Engineering Informatics*, 22, 71–89.

Stouffs, R., 2018. Implementation issues of parallel shape grammars. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 32, 162–176.

Stouffs, R. and Krishnamurti, R., 2019. A uniform characterization of augmented shapes. *Computer-Aided Design*, 110, 37–49.

Stouffs, R. and Rafiq, Y., 2015. Generative and evolutionary design exploration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 29, 329–331.

Stouffs, R., Tauscher, H., and Biljecki, F., 2018. Achieving Complete and Near-Lossless Conversion from IFC to CityGML. *ISPRS, International Journal of Geo-Information*, 7 (9), 355.

Stouffs, R. and Tuncer, B., 2015. Typological Descriptions as Generative Guides for Historical Architecture. *Nexus Network Journal, Architecture and Mathematics*, 17, 785–805.

Toth, B., *et al.*, 2012. Custom Digital Workflows. *Proceedings of the 17th International Conference on Computer-Aided Architectural Design Research in Asia*, 163–172.

Uggla, G. and Horemuz, M., 2018. Geographic capabilities and limitations of Industry Foundation Classes. *Automation in Construction*, 96, 554–566.