ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume IV-4/W8, 2019
14th 3D GeoInfo Conference 2019, 24–27 September 2019, Singapore

GRAPH TRANSFORMATION RULES FOR IFC-TO-CITYGML ATTRIBUTE CONVERSION

Joie Lim [1], Helga Tauscher [2,*], Filip Biljecki [1]

[1] National University of Singapore, Singapore – (joie.lim, filip)@nus.edu.sg
[2] Dresden University of Applied Sciences, Dresden, Germany – helga.tauscher@htw-dresden.de

KEY WORDS: BIM, GIS, IFC, CityGML, graph transformation, conversion, attributes, properties

ABSTRACT:

In model transformation, the population of attributes on the target side constitutes the last step of the conversion process that carries over that part of the input which is often perceived as the most valuable actual information. We are employing a graph-based model transformation approach to convert building information models into geospatial city models. In this paper, we are reporting on different types of transformation rules to populate the attributes on CityGML side using information extracted from the IFC data. We document the various ways how attribute values can be stored in IFC and CityGML respectively and identify patterns that bridge these endpoints in the conversion process. These patterns lead to a set of prototypical graph transformation rules which have been applied to a range of building projects. The novel graph-based approach to IFC-to-CityGML conversion implicates an intuitive visual representation of these rules. This work can also serve as a starting point to convert IFC data to other formats or to populate CityGML from other data sources.

1. INTRODUCTION

1.1 IFC-to-CityGML conversion

Interoperability between the domain of Architecture, Engineering and Construction (AEC) and the geospatial (urban) domain has become a highly-researched topic in the recent years (Liu et al., 2017; Stouffs et al., 2018; Ellul et al., 2018; Wang et al., 2019; Zhu et al., 2018). Information from Geographic Information Systems (GIS) and Building Information Modelling (BIM) supplement each other thus supporting cross-domain analysis such as view and shading analysis, tower crane location optimization (Rafiee et al., 2014; Irizarry, Karan). Benefits from digital modelling in both domains, e.g. improved analysis and simulation methods, continuous information flow across the lifecycle, reuse of existing datasets can be scaled up through integration of the two.

Industry Foundation Classes (IFC) and CityGML, an application schema of the Geographical Markup Language (GML), are the primary standards used for information transfer in the BIM and GIS domain respectively. Interoperability between the two is researched in a variety of ways, such as by looking at compatibility, overlaps and mismatches, between the respective conceptual models, by converting between IFC datasets and CityGML datasets or by integrating BIM and GIS datasources into larger systems (El-Mekawy et al., 2012; Gilbert et al., 2018).

While much effort has been put into the conversion of IFC to CityGML, many of the research projects focus on geometry reduction (generalisation) and geometry validity, especially in relation to creating models of different levels of detail and producing geometry that conforms to CityGML standards (Donkers et al., 2016; Kang, Hong; Floros et al., 2017). Not as many deal with the semantic data and attributes or properties present in the IFC models that could potentially inform or be helpful to downstream use of the converted models. Some that do this include Hor's approach using semantic web technology and RDF graphs and Deng's approach using reference ontology (Hor et al., 2016; Deng et al., 2016).

This paper aims to share an approach we have used as an important component of a project on the conversion from IFC to CityGML. It focuses on the properties portion of the conversion and rule development processes.

1.2 Graph transformation approach

We have taken a graph transformation approach to convert from the IFC data to CityGML. This approach takes the form of a triple graph grammar, a concept first introduced by Schürr (1995). Applied here, one graph represents the objects and relations on the IFC side, a second graph stands for objects and relations on the CityGML, and a third graph carries the correspondence between the two main graphs.

While a regular triple graph grammar can be operationalized for transformation in both directions as well as for synchronization of integrated systems, we focus solely on forward transformation (Stouffs et al., 2018) from IFC to CityGML. That means that the full IFC object graph is given as start graph together with an empty CityGML and connection graph. The latter two empty graphs are populated through out the application of graph transformation rules, effectively creating the CityGML graph from the given IFC input and the correspondence relations between the two.

1.2.1 Layered rule repository. In our implementation, we created a rule repository, made up of multiple modules consisting of rules that map different portions of the IFC schema to the CityGML schema. The scopes of the modules are for instance the general model structure, the spatio-semantic structure, the geometry and the properties.

* Corresponding author

https://doi.org/10.5194/isprs-annals-IV-4-W8-83-2019

Each rule defines an isolated part of the conversion logic, mostly to generate one type of object. The rules take the form of small triple graphs (template graphs) that define certain patterns and conditions that have to be satisfied in order to trigger the application of the rule. They also define the nodes that are to be created on the CityGML graph. Finally, they define how information is obtained, processed and stored, through the assigning of converters.

Based on the information present in the IFC file and the result desired for the CityGML file, relevant rules from each module can be selected and combined to create a rule set that generates the desired output. This rule set can then be used to perform conversions, repeatedly for different input data. Each single rule can be created and modified to process information in different ways for each particular object, including the combination of multiple properties or functions to further transform the values to conform to the CityGML schema.

This method allows a range of different options for customizing the IFC to CityGML conversion. Rule sets can be compiled for different scenarios, from converting entire IFC models to extracting floor plans of single storeys. Tauscher (Stouffs), for example, show different conversion variants with respect to the spatio-semantic paradigm.

1.2.2 Property rules. Forward transformation rules in the property module process non-spatial attribute values (information that is not related to either the semantic structure or geometry) from the IFC graph, transform them and add the results to the CityGML graph. These added properties supplement the semantic structure and geometry and provide additional non-geometric information for the various use cases the CityGML file might be used for.

Property rules assume that the semantic structure and geometry of the graph is already present. During the application of a property rule specific values are extracted from and added as attributes to matching nodes of the IFC and CityGML object graphs. The structure of these rules may differ depending on where these properties are to be found in the IFC graph and where they should be inserted in the CityGML graph.

1.3 ADE

Application Domain Extensions (ADE) are extensions to the standard CityGML schema, allowing objects and information that are not part of the CityGML schema to also be stored within a CityGML file. ADEs have been developed for a variety of purposes (Biljecki et al., 2018), including supporting the conversion between the two data models (de Laat, van Berlo; Deng, Cheng).

In our project, an ADE had been developed to allow more properties from IFC to be represented in CityGML as attributes. For example, we have extended the standard CityGML set of attributes for buildings and their components (e.g. wall material, building function according to extended code lists) and introduced new features (e.g. elevators) to accommodate some potentially useful information from IFC that cannot be stored in the CityGML data model by default. The ADE was designed based on analysing the requirements of multiple use cases.

In some of the examples in the paper, we use `ifc:IfcProperty`, an ADE element made to replicate the IfcProperty object from IFC. It a property name and a property value, both as Strings, and allows for a generic representation of arbitrary properties without explicitly adding each single required attribute to the ADE.

1.4 Implementation

The implementation has been developed as three parts. The first part is an instance of BIMserver, an opensource IFC database [1]. The second part is a web-based rule repository, where rules and rulesets are created and stored in a customised Domain Specific Language. These rulesets are then converted into JSON for use with the third part, which is a java client that retrieves models from BIMserver and handles the bulk of the conversion process.

Due to the added features that improve interoperability between the two file formats (Cheng et al., 2014; Kutzner, Kolbe), we have chosen to use IFC4 and an early developmental version of CityGML 3.0 in the development and testing of rules and the implementation.

1.5 Structure of the paper

The reminder of this paper is structured as follows: In Section 2 we describe different ways how properties and attributes appear in IFC and CityGML. In Section 3 we deduce 6 types of rules that cover the various possibilities on IFC and CityGML side in combination, and can be employed in many situations. In Section 4 we show the results of applying these rule to a use case.

2. PROPERTIES AND ATTRIBUTES IN IFC AND CITYGML

Properties and attributes are defined and expressed differently in IFC and CityGML, with a number of different possible ways in each. In order to identify common patterns that could be used to guide property rule creation, we explored how properties and attributes appear in IFC and CityGML respectively. This section shares our findings.

We have focused mainly on the architecture domain of IFC and the building module of CityGML, based on the models we were able to study as well as the desired CityGML output. However, the overall structure of the two schemas should be similar for other portions as well and the study could be expanded upon in the future.

2.1 Direct attributes in IFC

Some attributes in IFC are defined directly as attributes of entity types, mainly of `IfcObject` or `IfcType` subtypes. For example, the `name` and `longName` of an `IfcSpace` are defined within the definition of the `IfcSpace` itself, as the third and eighth attributes respectively. In the template graph of the rule, these are shown as part of the node, which must be of the respective type where the attribute is defined. An example excerpt from IFC is shown below.

```
#7605= IFCSPACE('06UQUgDKf1OOFHDJEkk1oK',#42,'160',
  $,$,#7340,#7602,'GYMNASIUM',.ELEMENT.,.SPACE.,$);
```

When developing rules and converters, it must be considered that some of the attributes defined in the IFC schema may be optional and thus hold indefinite values ($).

[1] Open source BIMserver: http://bimserver.org

https://doi.org/10.5194/isprs-annals-IV-4-W8-83-2019

2.2 Property sets in IFC

Other information not defined directly as attributes of entities can be defined in properties, for all kinds of information related to an entity, and quantities, for measured values like lengths and areas. These just contain additional information and are grouped into sets called property sets or quantity sets respectively. Properties, quantities as well as property and quantity sets are separate entities that consequently appear as separate nodes in the rules' template graphs.

Properties and property sets constitute a generic concept, that can be customized to hold any information one may need. However, the IFC schema contains a large number of predefined options for common choices, in order to establish interoperability. One example is the Pset_SpaceCommon property set that contains properties such as `HandicapAccessible` and `IsExternal`. An example excerpt is shown below.

```
#252= IFCSPACE('2Tz$9V8aP4DwHsKEQi1yWQ',#42,'1',$,
  $,#225,#247,'Area',.ELEMENT.,.SPACE.,$);
#273= IFCPROPERTYSINGLEVALUE('Reference',$,
  IFCIDENTIFIER('Area 1'),$);
#274= IFCPROPERTYSINGLEVALUE('IsExternal',$,
  IFCBOOLEAN(.F.),$);
#275= IFCPROPERTYSET('2$8eoYr41AHhjBt9l2GWS2',#42,
  'Pset_SpaceCommon',$,(#273,#274));
#283= IFCRELDEFINESBYPROPERTIES(
  '0bPQxpV8TDjQ_CEqzg9t7f',#42,$,$,(#252),#275);
```

In the template graph, these are expressed as a path of nodes, going from the node of the IFC object or type, to the property set or quantity set, to the property or quantity. The values are then direct attributes of the property or quantity nodes.

This kind of generic properties and property sets may be used and thus be referenced by multiple objects. For example, the `HandicapAccessible` property of `IfcSpace` has only 3 possible values: True, False, and Undefined. In many cases, it may only be defined once per possible value and each `IfcSpace` would reference the corresponding property definition. So even if multiple `IfcSpaces` are handicap-accessible, there would only be one `HandicapAccessible` property defined and it would be shared by all the `IfcSpaces`. Likewise, a whole property set can be shared among similar entities, for instance if some spaces are all accessible, as shown in Figure 1.

2.3 Direct attributes in CityGML

Most information in CityGML (including information that is found in most ADEs in practice) are simple values, like primitives or enumerations, expressed as either elements directly in the content or attributes of the city objects. Examples of this are id and name as shown below.

```
<bldg:BuildingRoom gml:id="ifc-2Tz-9V8aP4DwH...">
   <gml:name>1</gml:name>
   <!-- ... --->
</bldg:BuildingRoom>
```

Although these appear different in the XML-representation of CityGML, the way they are expressed in the template graphs are the same. Both are expressed as part of the object's node as shown in Figure 2.

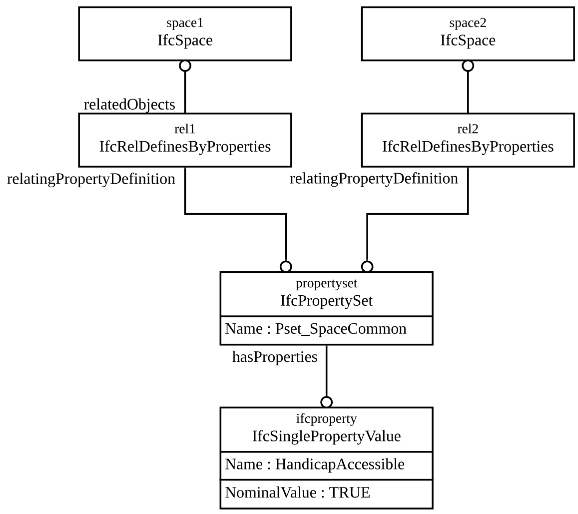

Figure 1. An instance of an `IfcPropertySet` node referenced by multiple entities in the IFC object graph.

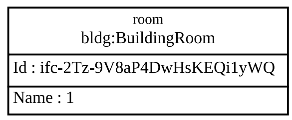

Figure 2. An instance of a `BuildingRoom` node in the CityGML object graph with some direct attributes.

2.4 Dedicated property elements in CityGML

In some instances when we are dealing with more complex or customized properties, they are defined CityGML or in the ADE as separate complex types instead of just simple attributes. These complex types have more than one attribute. An example is the address of a building. In the ADE we have defined a dedicated IfcProperty element to resemble the generic IFC properties, which also makes use of a similar construction as shown below.

```
<bldg:BuildingRoom>
  <ifc:ifcProperty>
    <ifc:IfcProperty propertyName="RoomType"
                     propertyValue="commonArea"/>
  </ifc:ifcProperty>
  <!-- more room attributes, geometry etc. --->
</bldg:BuildingRoom>
```

In the graphs, these complex properties appear as separate nodes to allow for the different values and attributes to be attached to them individually, as shown in Figure 3.

3. PROPERTY RULE PATTERNS

3.1 Overview of possible property rule patterns

The different possible configurations used to store information in IFC and CityGML as described in the previous Section 2 can be combined in a number of ways and result in the following possible rule structures:

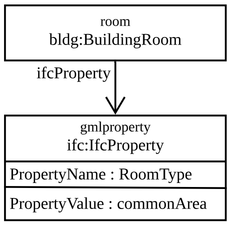

Figure 3. An instance of a `BuildingRoom` node with an `IfcProperty` node in the CityGML object graph.

1. From a direct attribute in IFC to a direct attribute in CityGML.
2. From a direct attribute in IFC to a new node in CityGML.
3. From a property in IFC to a direct attribute in CityGML.
4. From a property in IFC to a new node in CityGML.
5. From multiple sources in IFC to one attribute or node in CityGML.
6. From one attribute or property in IFC to multiple attributes or nodes in CityGML.

The rest of this section describes each of the above with examples and shares some limitations we encountered.

3.1.1 Direct attribute to direct attribute. Such rules extract a value directly from an existing IFC node and add an attribute directly to an already existing and connected CityGML node. With such rules, in our current implementation, if multiple rules apply to the same attribute on the same CityGML node, the values assigned by earlier rules are overwritten by subsequent rules. One example is the transfer of IDs from the IFC file into the CityGML file, as shown in Figure 4. This rule applies to all IFC entities of type `IfcProduct` corresponding to a CityGML object of type `AbstractGMLType`. The `GlobalID` attribute is then extracted from the `product`, transformed to conform to the constraints defined for IDs in GML in the `guid` function, and finally inserted as the `ID` attribute of the `gmlObject`.

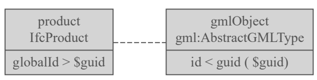

Figure 4. A rule converting a direct attribute to a direct attribute, here GlobalID in IFC to ID in CityGML.

3.1.2 Direct attribute to new node. Such rules extract a value directly from an existing IFC node, create a new CityGML node and add it to the CityGML graph by creating an edge between an existing CityGML node corresponding to the initial IFC node and the newly created CityGML node. One example is shown in Figure 5. This rule applies to an IFC node of type `IfcSpace` and an existing corresponding CityGML node of type `AbstractSpace` element. The name attribute, which many BIM software populate with the room's number during export [2], is extracted and inserted into the CityGML graph as a newly created `IfcProperty` node.

One issue encountered during an early implementation stage of the conversion client was that some rule types were not

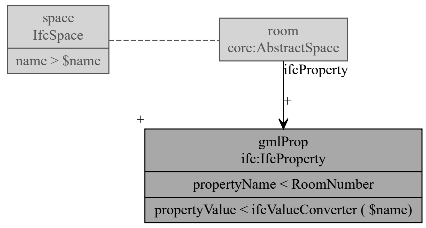

Figure 5. A rule converting a direct attribute to a new node, here the name attribute of IfcSpace in IFC to a generic IfcProperty node named RoomNumber in CityGML.

supported yet, e.g. rules with a newly created node on the CityGML side that does not have a corresponding node in the IFC side. A workaround that we employed at this stage was to forcefully detach the attribute on the IFC side to create a new node. This leads to a rule structure similar to the one described later in Section 3.1.4 and shown in Figure 6.

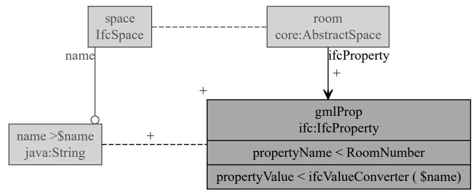

Figure 6. A rule similar to the one in Figure 5 working around an implementation issue.

3.1.3 Property to direct attribute. Such a rule extracts a value from a property or quantity that is located down a path from the entry IFC node and adds an attribute directly to the correlated existing CityGML node. With such rules, in our current implementation, if multiple rules apply to the same attribute on the same CityGML node, the values assigned by earlier rules are overwritten by subsequent rules.

One example is shown in Figure 7. This rule extracts the `IsExternal` property from the Pset_SpaceCommon property set of an `IfcSpace` typed entity and uses the value to determine the room's `spaceType`, assigning it to the respective `AbstractSpace` type `gmlElem` node in the CityGML graph.

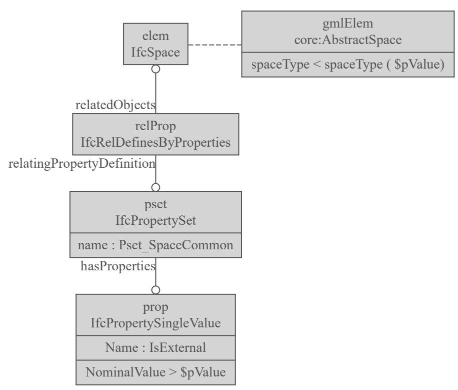

Figure 7. A rule converting a property to a direct attribute, here the IsExternal property of IfcSpace in IFC to the spaceType attribute in CityGML.

3.1.4 Property to new node. Such a rule extracts a value from a property or quantity that is located down a path from

[2] On a related note — for this piece of information it was necessary to extend the CityGML data model.

the entry IFC node, creates a new CityGML node and adds it to the CityGML graph by creating an edge between an existing CityGML node corresponding to the initial IFC node and the newly created CityGML node. Further it creates a correlation between the property nodes in IFC and CityGML.

One example is the FloorSurfaceMaterial rule, shown in Figure 8. This rule extracts the value of the IFC property with name `FloorCovering` from the Pset_SpaceCoveringRequirements property set of an IFC node of type `IfcSpace` and sets it as a new `IfcProperty` node of the corresponding existing CityGML node.

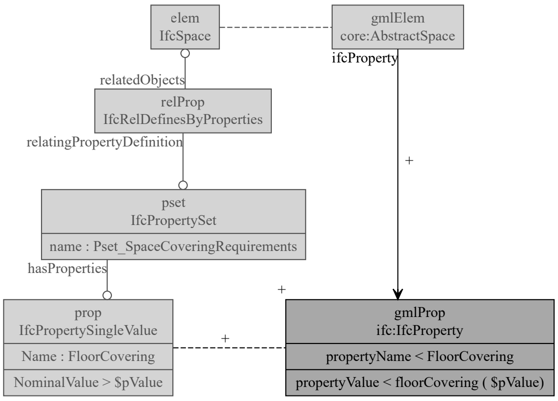

Figure 8. A rule converting a property to a new node, here the FloorCovering property of IfcSpace in IFC to an IfcProperty node.

3.1.5 Multiple sources to single target. These rules require multiple attributes and properties from the IFC graph to be combined in order to determine the desired value to be inserted into the CityGML graph. These come in the form of a number of different types.

One such type are rules that require attributes and properties specified in a different rule, and stored in the context during application of that other rule. One example of such a case is door height. In IFC, units can be omitted in measurement properties. In such cases, the properties take on the default units, defined on the level of the `IfcProject` node which is processed much earlier. To transfer such a property would require two rules. One rule to extract the units from IfcProject and store them in context, Figure 9, and one rule to extract that height value from the door, adjust it using the stored units, and assign it to a new node in the CityGML graph, Figure 10.

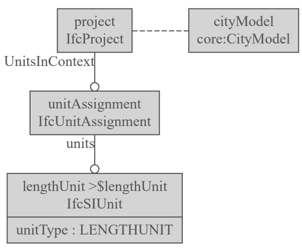

Figure 9. A rule extracting the default length unit from IfcProject to store it in the context variable $lengthUnit.

One issue with such rules is that the order in which the rules are applied is crucial. In this case, the first rule has to be applied before the second rule in order to work. In our implementation, we have worked around this by having rules

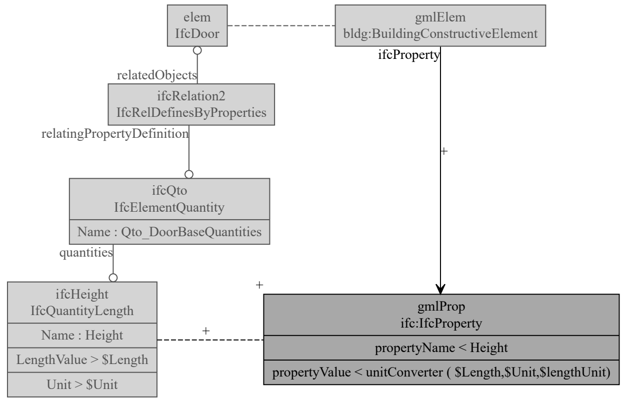

Figure 10. A rule extracting the height quantity from IfcDoor, converting it to meters using the $lengthUnit variable and setting it as a new IfcProperty node with the name DoorHeight in CityGML.

apply in alphanumerical order pertaining to their names, such that prefixes on rule names can be used to order them correctly.

Another type requires multiple of the same type of node to be used to determine the desired value. One example of such a case would be a rule for number of storeys above ground. This rule would require access to the `AboveGround` property for every storey before it can determine and assign a final value to the CityGML graph. With our implementation, in such a rule, each storey node would be processed individually. These types of rules would not be possible and would require workarounds instead.

3.1.6 Single source to multiple target. On the other hand, sometimes multiple attributes in the CityGML graph are derived from the same shared IFC node or attribute. These also come in a number of different types.

Some rules, as described in Section 2, involve a single, shared IFC node, referenced by multiple different parent nodes. One example is the rule for handicap access, Figure 11.

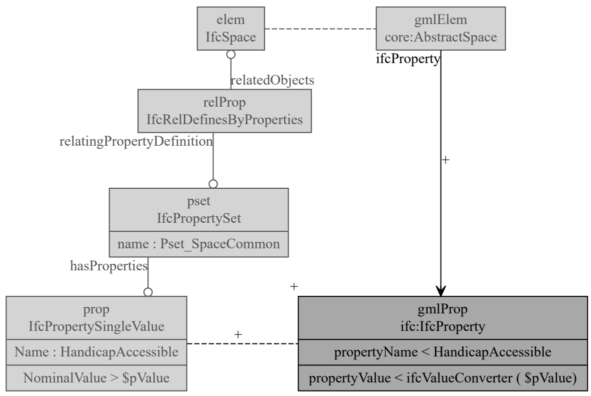

Figure 11. A rule that extracts the handicapAccessible property from IfcSpace and sets it as a new IfcProperty node in CityGML.

Because of checks to make sure each object is only converted once, there were issues with making sure that in such cases, the property is properly converted each time. In an early implementation stage, the conversion algorithm just refused to create a new CityGML node for a given node in the IFC object graph if there was already one. A more developed implementation now also checks whether there is an edge from the entry CityGML node to the potential duplicate and allowed creation of a second connected node if that is not the case.

Another type is when multiple attributes for the same node in the CityGML graph use the same attribute or property from the IFC graph. An example is the rules for `RoomName` and `RoomType`. Both rules require the `LongName` attribute of IfcSpace, as shown in Figure 12, and create new nodes. The IFC path for both rules are exactly the same. Hence, the current conversion implementation treats the second rule as a duplicate of the first and does not create the second property node.

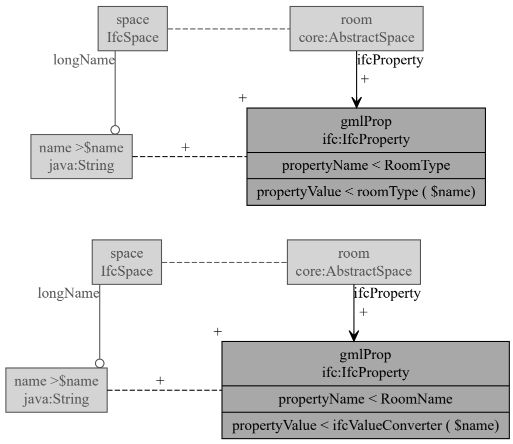

Figure 12. Rules that share the same left-hand side with nodes of types IfcSpace, java:String and core:AbstractSpace.

If the rules add attributes directly to the CityGML graph, they can simply be combined to work around this issue, as shown in the example in figure 13. However, if the rules create new nodes, the conversion client is unable to handle the combined rule.

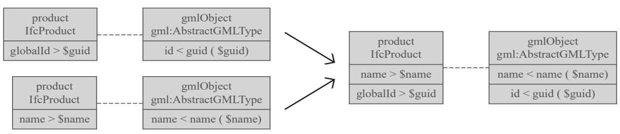

Figure 13. Rules for GML ID and GML name, combined into one rule.

3.1.7 Other issues. Apart from the different combinations of where the information is obtained from the IFC graph and added in CityGML graph described above, there are also some issues that stem from the semantic structure and geometric structure of the graphs.

One such issue occurs when trying to set attributes on nodes in the CityGML graph that do not have corresponding nodes in the IFC graph. One example involves the `solid` node in the rule shown in Figure 14. Here, we wanted to implement a rule that adds a randomly generated ID attribute to all AbstractGMLType nodes. However, because this rule creates nodes that do not correspond to the IFC graph and our conversion implementation is heavily based on pairs of an IFC and a CityGML node as nexuses between subsequently applied rules (entry and exit node pairs), we were unable to do so for these particular nodes.

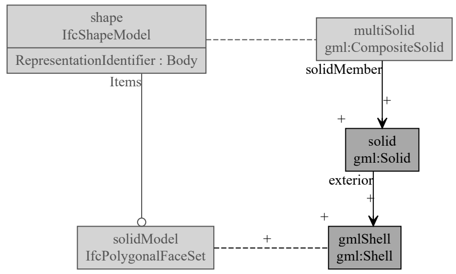

Figure 14. Rule that creates a node of type Solid which has no corresponding node in the IFC graph

4. RULE APPLICATION

4.1 Use cases

During our project on developing an automated approach to convert IFC to CityGML, we have considered various circumstances that would shape the development of our conversion mechanism. Besides examining specific datasets that were at our disposal and considering properties pertaining to the geographic area we are dealing with, the development of the conversion was driven by the intended use of the output (CityGML) datasets.

We have carried out discovery sessions together with users of the urban models, and identified a number of use cases for which the datasets would be in general suitable for use. We focused on three use cases that would be beneficial to look into more details, and identified relevant properties that would be potentially useful for these cases. The three domains of interest are namely indoor navigation, urban planning, and energy studies.

For each of the use cases we have highlighted a set of properties that should be preserved during the conversion from IFC, and for some of these the ADE had to be developed. For example, for indoor navigation, it was deemed beneficial to retain the information on elevators, stairs, accessibility of spaces, and size of doors.

4.2 Ruleset

The resulting ruleset contains a total of 21 property rules, distributed across the types described in Section 3. Some of the rules correspond to more than one type. The ruleset contains

- 2 rules of type 1,
- 3 rules of type 2,
- 1 rule of type 3,
- 15 rules of type 4,
- 6 rules of combination type 5,
- 4 rules of combination type 6.

Conversions were carried out with the created ruleset on a number of files. Three of them have been selected and studied below, a full IFC model of a non-residential building in Singapore we have been provided with (BCA Academy) and two residential estates (A and B) exported from Revit and ArchiCAD files respectively. The resulting CityGML files, generated with our conversion engine, are shown in Figure 15.

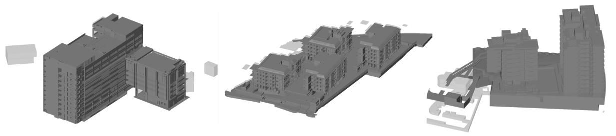

Figure 15. Visualisation of resulting CityGML files from the BCA Academy dataset (left), Estate A dataset (center) and Estate B dataset (right).

Through the conversion, we also tracked how many times each individual rule and each rule type was applied. Table 1 shows the total number of times each rule type is applied for each IFC file. Table 2 shows the details of each individual rule, its rule type and how many times they are applied for the two IFC files.

	BCA Academy	Estate A	Estate B
Rule Type 1	222410	325601	40664
Rule Type 2	1415	4506	939
Rule Type 3	701	2247	0
Rule Type 4	23150	62086	9709
Rule Type 5	2094	5773	0
Rule Type 6	1415	3566	1

Table 1. Number of applications for each rule type in the 3rd storey model and the full BCA Academy model

5. CONCLUSIONS

5.1 Discussion

The study has identified what and how attribute information is represented in IFC and CityGML, and how they differ. This helped us to gain a better understanding of the different factors that affect how this type of information can be transferred from IFC to CityGML. We have also described some of the issues and difficulties that we encountered during the development of property rules for our conversion application based on graph transformation.

Having developed a rule set that successfully converted the properties we have identified, it has also demonstrated the possibility and flexibility of using a graph transformation approach for property conversion. We managed to transfer 19 different properties across the 6 different scenarios we have identified.

Through this process we have also managed to identify more nuanced scenarios that added complexity to the graph transformation approach for mapping information from one file format to another. Towards the larger efforts of developing a triple graph grammar approach for conversions between any two file formats, knowledge of these scenarios has helped to gain a better understanding of what might be required for a more robust and sufficiently flexible conversion procedure.

5.2 Limitations

Due to the way the property rules have been separated into individual graphs for each property, combining information from different parts of the file is difficult. Many of the rules of type 5 and 6 require unintuitive workarounds or are not possible at the current stage of our implementation.

Also, the properties and attributes we have studied are only a subset of what is possible in the IFC and CityGML schemas, based on the models we were able to access and what information might be useful for the use cases we considered. With further studies of other portions of the two schemas (e.g. the electrical or construction management domains for IFC and the transportation or vegetation modules for CityGML), it is possible that other cases and nuanced scenarios may be found. These may result in additional rule structures.

5.3 Future work

Other possibilities for future work could lie in how these rules could be developed more quickly or with less manual work based on the patterns we have identified. It would be useful if the patterns could be used to generate rules automatically given a simple list of property names or the schema definitions.

Another possibility could involve studying more file formats used in the building and geospatial community. It may be possible that some larger or more generic patterns could be derived from how information is stored differently across all of them.

ACKNOWLEDGEMENTS

We gratefully acknowledge the comments from the reviewers, and the IF2CityGML project team. This material is based on research/work supported by the National Research Foundation under Virtual Singapore Award No. NRF2015VSG-AA3DCM001-008.

REFERENCES

Biljecki, F., Kumar, K., Nagel, C., 2018. CityGML Application Domain Extension (ADE): overview of developments. *Open Geospatial Data, Software and Standards*, 3(1), 13.

Cheng, J. C. P., Deng, Y., Das, M., Anumba, C., 2014. Evaluation of IFC4 for the GIS and green building domains. *Computing in Civil and Building Engineering*, 2216–2223.

de Laat, R., van Berlo, L., 2011. Integration of BIM and GIS: The development of the CityGML GeoBIM extension. *Advances in 3D Geo-Information Sciences*, Springer Berlin Heidelberg, 211–225.

Deng, Y., Cheng, J. C., 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.

Deng, Y., Cheng, J. C. P., 2015. Automatic Transformation of Different Levels of Detail in 3D GIS City Models in CityGML. *International Journal of 3-D Information Modeling*, 4(3), 1–21.

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

El-Mekawy, M., stman b, A., c, I. H., 2012. An Evaluation of IFC-CityGML Unidirectional Conversion. *International Journal of Advanced Computer Science and Applications*, 3(5).

Ellul, C., Stoter, J., Harrie, L., Shariat, M., Behan, A., Pla, M., 2018. Investigating the state of play of GeoBIM across Europe. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XLII-4/W10, 19–26.

Rule Name	Rule Type	Combination	No. of application (BCA Academy)	No. of application (Estate A)	No. of application (Estate B)
IsExternal2SpaceType	3	6	701	2247	0
HandicapAccessible	4	6	0	0	0
Internal & External Access	4	6	703	1570	1
FloorSurfaceMaterial	4		56	0	0
Name2RoomType	4		701	2247	462
MaterialLayerSet2Material	4		12364	31771	0
MaterialConstituentSet2Material	4		1912	3044	0
NoOfStories	4		1	1	1
RoofTilt	4	6	11	16	0
RoomHeight	4	5	701	2247	0
RandomUuid2Gmld	1		205257	227318	27104
IfcGuid2GmlId + GmlName	1		17153	98283	13560
RoomName	2		701	2247	462
StoreyName	2		13	12	15
DoorWidth	4	5	567	1570	0
RoofArea	4	5	11	204	0
WindowHeight	4	5	122	91	0
WindowWidth	4	5	122	91	0
DoorHeight	4	5	571	1570	0
RoomNumber	2		701	2247	462
ThermalTransmittance	4		5308	17664	9243

Table 2. Table of each rule, its corresponding type and how often each rule is applied in the BCA Academy Model, Estate A Model and Estate B Model

Floros, G., Pispidikis, I., Dimopoulou, E., 2017. Investigating integration capabilities between IFC and CityGML LOD3 for 3D city modelling.

Gilbert, T., Barr, S., James, P., Morley, J., Ji, Q., 2018. Software Systems Approach to Multi-Scale GIS-BIM Utility Infrastructure Network Integration and Resource Flow Simulation. *ISPRS International Journal of Geo-Information*, 7(8).

Hor, A.-H., Jadidi, A., Sohn, G., 2016. BIM-GIS Integrated Geospatial Information Model Using Semantic Web And RDF Graphs. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-4, 73-79.

Irizarry, J., Karan, E. P., 2012. Optimizing location of tower cranes on construction sites through GIS and BIM integration. *ITcon*, 17, 351-366.

Kang, T. W., Hong, C. H., 2017. IFC-CityGML LOD mapping automation using multiprocessing-based screen-buffer scanning including mapping rule. *KSCE Journal of Civil Engineering*, 4(4), 1–11.

Kutzner, T., Kolbe, T. H., 2018. CityGML 3.0: Sneak preview. *38. Wissenschaftlich-Technische Jahrestagung der DGPF und PFGK18 Tagung*, Publikationen der DGPF, 27, Munich, Germany, 835–839.

Liu, X., Wang, X., Wright, G., Cheng, J., Li, X., Liu, R., 2017. 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.

Rafiee, A., Dias, E., Fruijtier, S., Scholten, H., 2014. From BIM to Geo-analysis: View Coverage and Shadow Analysis by BIM/GIS Integration. *Procedia Environmental Sciences*, 22, 397 - 402.

Schürr, A., 1995. Specification of graph translators with triple graph grammars. *Graph-Theoretic Concepts in Computer Science*, 151–163.

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

Tauscher, H., Stouffs, R., 2019. Extracting different spatio-semantic structures from IFC using a triple graph grammar. *Intelligent and informed: 24th Annual Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA 2019)*, Wellington, New Zealand, 605–614.

Wang, H., Pan, Y., Luo, X., 2019. Integration of BIM and GIS in sustainable built environment: A review and bibliometric analysis. *Automation in Construction*, 103, 41–52.

Zhu, J., Wright, G., Wang, J., Wang, X., 2018. A Critical Review of the Integration of Geographic Information System and Building Information Modelling at the Data Level. *ISPRS International Journal of Geo-Information*, 7(2).