VISUALISING DETAILED CITYGML AND ADE AT THE BUILDING SCALE

J. Lim [1], P. Janssen [1], F. Biljecki [1,2]*

[1] Department of Architecture, National University of Singapore, Singapore
[2] Department of Real Estate, National University of Singapore, Singapore

Commission VI, WG VI/4

KEY WORDS: CityGML, ADE, visualisation, 3D city models, BIM

ABSTRACT:

There is an increasing activity in developing workflows and implementations to convert BIM data into CityGML. However, there are still not many platforms that are suitable to view and interact with the detailed information stored as a result of such conversions, especially if an Application Domain Extension (ADE) is involved to support additional information. We investigated the ease of use and features supported by visualisation software and tools with CityGML and ADE support, and propose an approach to develop a tool that combines useful features using a set of generic rules that can extract CityGML ADE attributes. The work, while generic, is geared towards detailed architectural datasets sourced from BIM. We implemented the approach in a web-based viewer supporting the visualisation of CityGML datasets enriched with ADE features.

1. INTRODUCTION

There has been an increasing effort into converting Building Information Modelling (BIM) datasets into CityGML (El-Mekawy et al., 2012; Liu et al., 2017; Zhu et al., 2018; Noardo et al., 2019). One of the main motivation is to leverage on information created during the design phase of construction (Donkers et al., 2016; Stouffs et al., 2018), enabling the generation of highly detailed models with interior benefiting many applications (Boeters et al., 2015; Isikdag et al., 2013; Chen et al., 2014; Atazadeh et al., 2017; Biljecki et al., 2015).

Such conversions have been accomplished by different methods, some of which simplify and omit information from the BIM models to better suit CityGML and the geospatial world (Deng et al., 2016; Donkers et al., 2016; Kang and Hong, 2017; Lim et al., 2019). These methods sometimes extend the CityGML schema using Application Domain Extensions (ADEs) to accommodate the rich information found in BIM models (de Laat and van Berlo, 2011; Biljecki et al., 2018). While such approach aids preserving the rich information, a downside is that CityGML datasets enriched with ADEs usually cannot be appropriately visualised and worked with, using the currently available CityGML software and tools.

This paper has multiple goals. First, we provide a brief introduction to CityGML and ADE, as well as a review of software and tools available to support the visualisation of and interaction with CityGML files containing ADEs (both from the geometric and semantic aspect), focusing on their ease of use and the features they provide for interaction with objects in the model and their attributes. Since building-scale data may be heavy, we also include large files in the review.

Second, we discuss briefly the fact that there are few options that allow such interaction to be done easily and appropriately for these detailed building models obtained from BIM models, we suggest and demonstrate a possible approach to develop such a tool. This tool would ideally be able to open and visualise any CityGML model containing ADE attributes, support viewing of all attributes linked to any selected object, and support filtering of the objects in view by any of the detected attributes, with the possibility of adding more features in the future. This would all be done with a minimum amount of setup, both for getting the platform ready and preparing the models for viewing, such that models can be viewed at the click of a button. The developed approach uses a combination of generic rules to extract information from both standard CityGML and ADEs, and by utilising a flexible web interface that has a wide support community. We hope that such a tool would be able to offer some support towards development and usage of other tools and ADEs at the building scale. Finally, we evaluate the limitations of our approach and briefly discuss other possible avenues for exploration in future work.

* Corresponding author at filip@nus.edu.sg

2. CITYGML AND ADE

CityGML is a data model for 3D city models standardised by the Open Geospatial Consortium (OGC). It is based on Geography Markup Language (GML) and it consists of a number of modules that support different aspects of the urban environment, including buildings, city furniture, vegetation, and terrain (Open Geospatial Consortium, 2012). One of the strong points of CityGML is its ability to store and communicate a large amount of semantic information in its models, in addition to visual and geometric information (Gröger and Plümer, 2012). For example, this includes information on surface types, heights, addresses, and names.

In addition to the wide range of semantic information it covers, CityGML features the Application Domain Extension (ADE) mechanism. These are additional, user-defined schemas that extend the standard CityGML to handle information aimed at specific domains. ADEs extend CityGML in different ways, e.g. geometrically (adding new geometric types), and semantically (adding new types of attributes and relations) (van den Brink et al., 2013; Kumar et al., 2018). A large number of ADEs defined for different uses have been created, documented, and researched (Biljecki et al., 2018). Some, such as the NoiseADE and EnergyADE (Czerwinski et al., 2006; Agugiaro et al.,

2018), have become widely used and some have been recognized and added as additional thematic modules to the native CityGML schema, e.g. Bridge ADE with CityGML 2.0 and the Dynamizer ADE with the upcoming CityGML 3.0 (Chaturvedi and Kolbe, 2016; Kutzner et al., 2020).

3. REVIEWING CITYGML VISUALISATION AND ADE SUPPORT

There is a number of visualisation and geospatial software that support CityGML. However, many use cases involve using and visualising CityGML models by converting them to other data formats, since it is primarily intended as a data exchange model. This section reviews relevant software packages that are able to read CityGML data and visualise them either directly or through their own pipeline, without needing users to convert files separately using other software before viewing. Our review is focused on understanding their capabilities to read and display information provided through the ADE.

3.1 Methodology

We looked at seven software packages that are currently available, by opening and viewing six different datasets, focusing on how much of the files' geometry and properties the programs were able to support and view. This approach is similar to the methodology of Noardo et al. (2019), with our review more focused on ADEs. We also noted how easy it is to open a file for viewing and view the properties of the different objects in the model. Our selection of datasets includes a large CityGML model of Rotterdam in LOD1 (3D Geoinformation group, 2019), a NoiseADE sample dataset for buildings provided by OGC (Open Geospatial Consortium, 2012), an EnergyADE sample dataset of a house (KIT, 2017), and a UtilityNetworkADE sample dataset of the water pipes (Boates, 2019). The remaining two files were developed by our team, both sourced from a conversion from IFC – one containing indoor data in the standard CityGML form without an extension (Konde et al., 2018), and another of a large building containing an ADE. An overview of the files and their contents are shown in Table 1.

3.2 Software descriptions and observations

3.2.1 SpiderViewer SpiderViewer is a CityGML viewer developed by GEORES. It allows the customising of appearances and textures based on the semantic properties of CityGML elements. In our tests it was able to detect ADE schemas and namespaces within the files but was unable to resolve the schemas and thus was unable to display these models.

3.2.2 FZKViewer 5 One example of visualisation software that supports both CityGML and ADE is FZKViewer, developed by IAI/KIT, a tool for Windows that is able to visualise geometric and semantic information from IFC, gbXML, and CityGML files (KIT, 2020). In addition to the 3D view of the model, it is able to extract the hierarchy and relationships between objects in the file, as well as tables of properties or statistics regarding the objects found. Out of the box, it can view geometries and semantic information from some selected ADEs with built-in support, such as the Energy ADE. It is also able to display the geometry and a portion of the properties for the NoiseADE but was unable to for the UtilityNetworkADE. It allows the user to navigate through the hierarchy of the file and hide objects with a simple interface. It also allows for interaction with individual objects, like individual wall surfaces or windows. Selecting an object in the 3D view or in the attributes table highlights it in all other windows and pulls up a list of properties for the selected object.

3.2.3 azul Azul is a tool for macOS that supports 3D visualisation of CityGML, CityJSON, OBJ, OFF and POLY files (Arroyo Ohori, 2020). It is able to open multiple files, and displays the attributes of objects when they are selected in the viewer or in the sidebar. It supports the IMGeo ADE and is able to extract geometries from any ADE for visualisation, but it has no full support for the extraction of ADE attributes. It was able to open files 1, 3, 5, and 6. With file 4, it was able to detect the geometry and list them in the sidebar, but no geometry was displayed in the viewer. With the files we used, some properties such as name and type were displayed but ADE attributes were not able to be extracted.

3.2.4 FME 2019.2 Another software package that supports CityGML and also comes with ADE functionalities is FME. It is a data integration tool and focuses on the conversion and translation of one format to another, supporting hundreds of different file formats and software, one of which is CityGML. It reads the XML file to find the ADE specifications where needed, or XSDs can be directly input to allow for ADE support. It comes with a visualising component, FME Data Inspector, which allows for the visualisation of information alongside the conversion process (Biljecki et al., 2018). It was able to open files 1, 2, 3 and 5 without any configurations or XSDs, with all NoiseADE properties from file 2 and EnergyADE properties and geometry from file 3. Files 4 and 6 require additional configuration in order to be opened.

3.2.5 QGIS 3.12 QGIS is an open-source geographic information system software that is able to load, view and edit CityGML files, among other supported file formats. When loading a CityGML file, objects from the different CityGML modules are detected and divided into different layers from which the user may select which ones they wish to import. It supports viewing the files in 3D, but this works only for certain types of files, while others may require pre-processing first. For example, while parts of the Rotterdam example are viewable in 3D, the NoiseADE example is only viewable in 2D. However, even though it is possible, we observed that viewing files in 3D can be taxing on the computer. ADE properties are fully imported and viewable in its property tables but ADE geometry may not be supported. For example, the FZKHouse example file's properties are all imported and can be viewed but the geometry is empty, while the Nanaimo water pipes file's properties and geometry are both supported.

3.2.6 CesiumJS and Cesium ion CesiumJS is an open-source JavaScript library for creating 3D geospatial visualisations. It supports a large variety of 3D file formats, such as glTF, and also offers a separate service, Cesium ion, that converts various other file types, including CityGML, to 3D tiles and hosts them on their server for viewing (Amato, n.d.). Currently Cesium ion is focused on providing efficient tiling and other options like textures and clamping to terrain. It is able to extract properties and store them in batch tables to be used in CesiumJS viewers, such as for viewing when objects are selected, but it does not support ADEs at the moment. Due to the conversion process, it treats buildings as single objects, so it is not possible to retain and view properties for individual building elements using this service. Cesium ion is easy to use and

Table 1. List of datasets used in our research.

No.	File	Version	ADE	Size
1	Rotterdam_CityGML.gml	CityGML 2.0	-	689056 kB
2	noiseADE_building_example.gml	CityGML 2.0	NoiseADE 2.0	7 kB
3	FZKHouseLoD3-ADE.gml	CityGML 2.0	EnergyADE 1.0	150 kB
4	nanaimo_water_pipes.gml	CityGML 2.0	UtilityNetworkADE 0.9.2	12994 kB
5	Two_storey_residential_building_sample-xxx-003-006.gml	CityGML 2.0	-	135 kB
6	BCA_Academy-004-002-002.gml	CityGML 3.0	IfcADE	18989 kB

converts and tiles selected files without any complicated configurations. Upon conversion, a link to a simple sandbox to preview the converted file is provided, but in order the view the models and its properties properly, customisation of this code or a separate CesiumJS implementation is required.

3.2.7 3DCityDB and 3DCityDB web-map-client Due to the large nature of urban-scale 3D models, a common way to manage CityGML datasets is the use of a database to store and manage the large amount of information. Often, in conjunction with this, is the visualisation of the information in a separate application or web-based interface, with models and data streamed as needed from a server. In such environments, the CityGML models are usually converted into other file formats with better visualisation support, and the attribute information, such as names, heights, and addresses, are extracted from the models and stored in a separate format to be queried as needed. However, the models themselves no longer carry the semantic information, which may be crucial for use cases. Such an approach has been researched with a wide variety of combinations between file formats and platforms such as JSON files with WebGL browser interfaces, custom C3D files with a Java applets, glTF with CesiumJS, and OBJ with mobile applications (Gesquière and Manin, 2012; Prandi et al., 2015; Simoes et al., 2015). A widely used database is 3DCityDB, an open source spatial relational database which employs either Oracle or PostgreSQL to store and manage the information found in CityGML files while retaining both semantic and geometric data (Yao et al., 2018). It provides a frontend interface which makes it easy to import CityGML files and convert and export information to KML, COLLADA, glTF and spreadsheets, automatically tiling the models in the process. It also offers a separate web client based on CesiumJS that can be configured to view the exported files in a web browser. In this web-map-client, different sets of files can be organised into different layers, objects can be hidden and revealed, shadows and terrains can be toggled, and it can also be set up to display object attributes upon selection (Figure 1).

Since v4.0.0, it also has implemented ADE support into their database, in which XSDs can be used to configure the database to store the required information. However, the ADE functionality is not yet fully automatically integrated with their frontend interface and will require the user to develop the explicit mapping information themselves in order to configure the database and process the models for viewing (Yao et al., 2018). A large number of development and research efforts into CityGML visualisation focuses on how to best use 3DCityDB to perform and look better, and integrate with other information sources and types. This includes development of different ways to display models, integration with services to draw data from a wide variety of sources and many other works (Prieto et al., 2012; Pispidikis and Dimopoulou, 2016; Wang et al., 2018). Due to the conversion of files to glTF and KML, the 3DCityDB web-map-client is only able to select what 3DCityDB calls 'top-level elements'. This means that it can only select entire buildings as one object and is not able to allow selection of individual building elements. Aside from this main disadvantage, even though the 3DCityDB interface greatly simplifies the setting up and managing of CityGML files, it still requires an amount of additional work to set up, configure and export the files and spreadsheets for viewing. In addition, although the web client supports the viewing of attributes upon selection, the built-in setup is done such that it requires the spreadsheets to be made publicly accessible via Google Fusion Tables. It would require a separate implementation to support this attribute viewing feature differently.

3.3 Results and takeaways

The results of opening the six files in the seven visualisation software are summarised in Table 2, while Table 3 shows a summary of the features that are supported. The main takeaway from these simple tests with a number of different software and files was that CityGML viewers that support ADE fully out of the box are not common. The ones that are able to do so require setting up with the ADE XSDs and other additional configurations to properly handle files containing ADEs. Meanwhile, those that are easy to use are only able to support ADEs partially. In addition, the options that rely on conversion of the CityGML into other formats are only able to treat buildings as single items and selection of individual building elements is not supported. When such a system is used with the detailed LOD3 or LOD4 models of buildings at the building-scale, the rich information stored deeper in the model, such as at each storey or in each room, cannot be accessed. Such a shortcoming can be detrimental for certain situations and use cases.

3.4 Lessons learned and goals for our work

Our project aims to look into ways to support the visualisation and querying of CityGML objects at a building scale, inclusive of ADE support, in order to view such models in an easy and meaningful manner. In addition, we also hope to devise a method to view CityGML models based on the CityGML 3.0 schema and customised ADEs that are still under development and do not have stable XSDs. After examining the features of existing software and tools and the types of information found in CityGML models obtained from BIM models, we believe the ideal outcome would be to have a viewer with the ability to read and display files using generic rules that extract both geometry and properties, hopefully with results similar to QGIS where all the ADE properties are automatically detected. These rules would be loosely based on the CityGML schema but will not have to rely on the schemas for the ADEs, such that no configuration or stable XSDs are needed. The viewer would have to be able to allow the user to select individual building elements and view their properties, much like FZK Viewer, in order to properly capture the all the information stored. We also hope that the viewer would have the accessibility, interactivity and

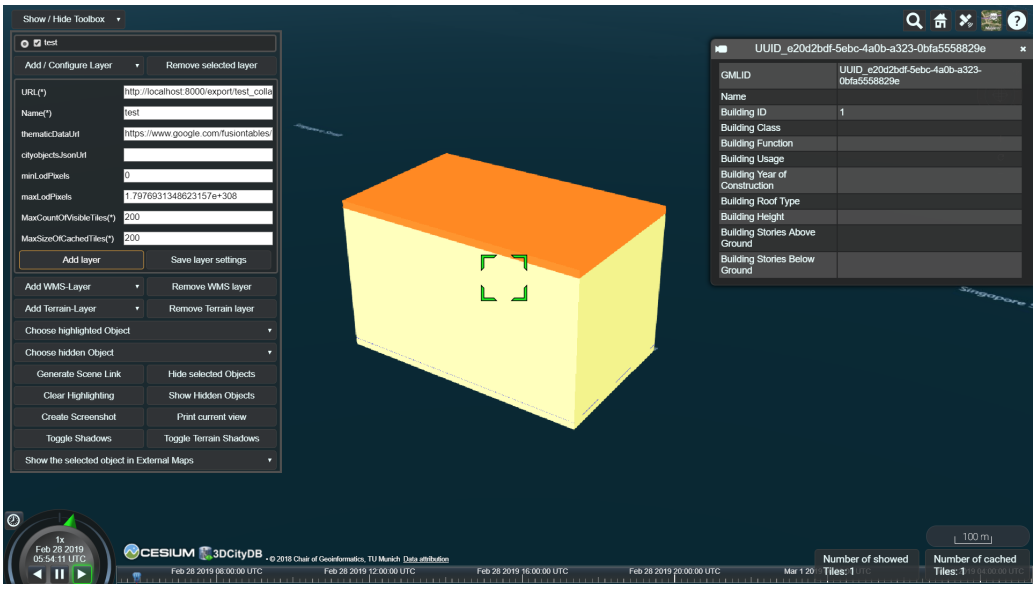

Figure 1. File 5 viewed using 3DCityDB web-map-client. It is possible to set it up to display a set of selected CityGML properties.

Table 2. Results of opening the files in the reviewed visualisation software.

	File 1	File 2	File 3	File 4	File 5	File 6
SpiderViewer	No	No	No	No	No	No
FZKViewer	Yes	Yes, partial ADE	Yes	No	Yes	Yes, no ADE
Azul	Yes	Yes, with issues	Yes, no ADE	No	Yes	Yes, no ADE
FME	Yes	Yes	Yes	Yes	Yes	Yes, with configuration
QGIS	Yes	Yes, 2D	Yes, no geometry	Yes	No	No
Cesium Ion	Yes	Yes, no ADE	No	No	No	No
3DCityDB	Yes	Yes, no ADE without significant configuration	Yes, no ADE without significant configuration	No without significant configuration	Yes	No

possibility of wider support for different information that the platforms created with 3DCityDB and CesiumJS offer. This would be done by detecting the native CityGML information in the file and extracting geometry and attributes according to the specified schema, while applying generic text content extraction for anything outside of the schema, which is assumed to be ADE information. In this way, the viewer does not need detailed information on the ADE and can be used to view ADEs still under development, or without XSD files. It could be a possible way to look at such files to get a better idea of what ADE properties are inside them, without having to look directly at the XML code.

4. METHODOLOGY

We have developed a viewer to demonstrate the implementation of a tool that would be able to read and display CityGML files with ADEs using three generic rules. This was done in the form of a web-based interface, using CesiumJS to support the visualisation.

4.1 CesiumJS and its Entity API

Apart from CesiumJS's support for a wide variety of file formats directly and through Cesium ion, it is also possible to create polygons and attach attributes to them on the fly, using its entity API. This allows us to read the CityGML files directly and create each polygon in the CityGML model separately. This would also allow each object or surface in the file to be viewed and selected as an individual object with its own semantic information, which is not possible with the other methods that use glTF or 3D Tiles. These include information such as its surface type and other attributes attached to the object, or information attached to objects higher in the tree such as information on the storey or building the object is located within. While not the most efficient solution in terms of performance, it is a simple method that could be used to view small files directly in the browser without the added hassle of setting up separate databases and servers or converting the file to other file formats.

4.2 Rule implementation process

The rule implementation process we have used is split into three parts. It aims to implement the rules in a way that will not require the entirety of the CityGML file to be checked against the schema, or keep too much of the file's hierarchy in memory while still allowing as much of the attributes to be extracted as possible. The first part is the identification of generic rules that would apply to a wide range of CityGML and ADE elements and allow a sufficient amount of information to be extracted properly with a small number of rules. The second part is the processing of the CityGML schema to derive a representation of these rules that could be used to inform the viewer of when each rule should be applied. Ideally, this would cover the entirety of the CityGML schema. The last is the development of a viewer that would apply these rules to the CityGML files it receives and extract the necessary information and display them in the

Table 3. Summary of the features supported by the considered software.

	Large files	ADE support	Ease of use	Select building elements
SpiderViewer	No	Partial	Drag and drop	No
FZKViewer	Yes	Partial	Drag and drop	Yes
Azul	Yes	Partial	Drag and drop	No
FME	Yes	Yes, may require configuration	May require configuration/setup	Yes
QGIS	Yes, but 3D is taxing on machine	Partial	Drag and drop	Yes
Cesium Ion	Yes, streamed	No	Requires configuration/setup	No
3DCityDB	Yes, streamed	Partial, requires configuration	Requires configuration/setup	No

desired manner.

4.2.1 Identification of rules For demonstration purposes, we have focused on the CityGML Building module and the Noise ADE, and identified 3 generic rules that can be used to extract information sufficiently to handle simple CityGML files and ADE attributes (however, our work is intended to be generic and applicable to other modules and ADEs as well): (1) Process for attributes, (2) Process for geometry, and (3) Check for geometry and attributes.

The first rule is used for elements that would directly lead to only textual information that should be grouped and displayed together. This rule is assigned to elements such as xalAddress and is the default rule used. This is also the rule used for ADE elements. The second rule is used for elements that would directly lead to only geometrical information and no longer contain any attributes that would be useful to be viewed as plain text. Such elements include lod4MultiSurface and lod1Solid, as well as their children. The third rule is used when an element could contain both geometrical and textual information in its children nodes. As such, they need to be processed further to determine what elements are further down the branch and what rules to apply. This rule is assigned to any element that would eventually lead to elements with rule 2. The combination of these three rules would allow simple geometrical and textual information to be extracted from the CityGML file in an appropriate manner to be displayed in the implemented viewer. This is just a subset of all the cases present in the full CityGML schema and is limited in its ability to handle different types of ADE schemas but we found it sufficient for the purposes of this demonstration.

4.2.2 Representation of rules After identifying the generic rules, the viewer would need specific instructions on when each rule should be applied. This was done by processing the CityGML XSDs to extract a JSON file of key-value pairs that instruct the viewer on what action to perform when a particular element is encountered in the CityGML file. The intention was to generate this file for any version of, or any subset of modules for CityGML as needed and specify it in the viewer. To achieve this aim, a recursive function is used to check each element in each of the relevant XSD files, following paths to all its possible substitution groups, types, references, extension bases and child nodes, and subsequently all possible paths from each one of those elements. When an element that matches a rule is found, a key-value pair is set and a value is returned to inform the determining of rules for parent elements as the function unwinds. Figure 2 illustrates this process using a flowchart.

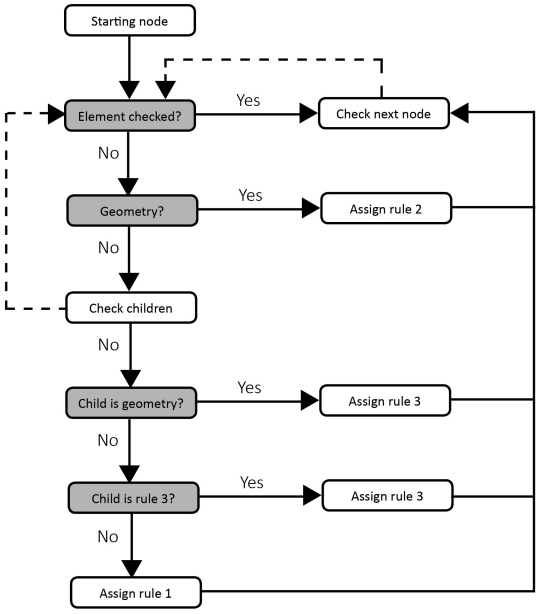

Figure 2. Illustration of the developed rules.

4.2.3 Application of rules We implemented the three rules from the first step, as well as some other rules that extracted information for positioning the models from detected EPSG codes, in the viewer. The rule application process is shown in Figure 3 using file 2 as an example.

On the left is an abstraction of the XML found in the file, which each element colour coded according its respective rule. Elements that should be processed with rule 1 are shown in light grey, rule 2 in black and rule 3 in dark grey. ADE elements which no rule matches are shown in white. The file is read starting from the cityModel element and it is checked against the JSON file generated during the second step to determine which rule should be used before moving on to its child elements. Matches for rule 1 are processed immediately. The element's and all subsequent child elements' values are all extracted and added to a javascript object as shown on the right. Elements that are unable to find a matching rule are also processed immediately and are added to a separate category for ADEs. This was done so we could better inspect the different properties in the file. Matches for rule 3 are processed next. Elements that match this rule are collected and checked after all its sibling elements have been checked. Newly extracted values are added to the javascript objects from before, creating new categories for each new hierarchy as required. Finally, matches for rule 2

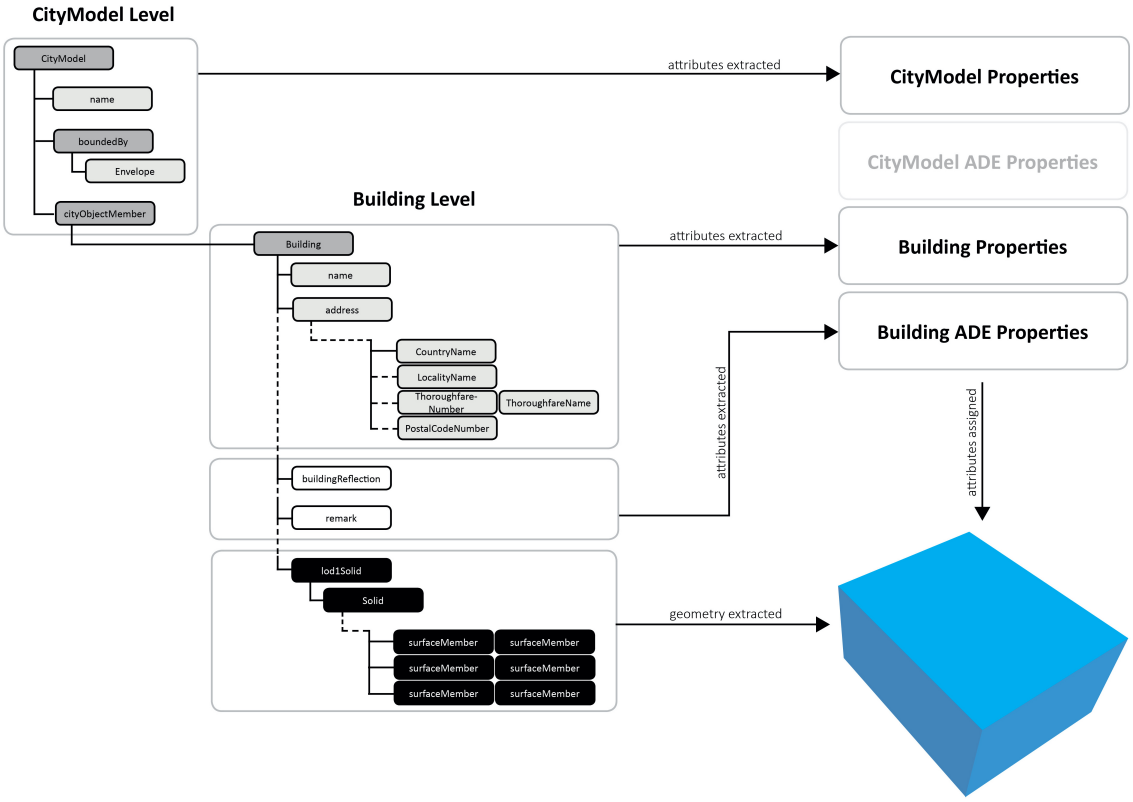

Figure 3. Rule application process.

are processed, after all the elements for rule 3 have been fully processed. The elements are processed to generate geometry and the most updated javascript object containing all the properties extracted is assigned to the Cesium entity created. In this example there was no geometry at the CityModel level but at the Building level this rule is used on lod1Solid, to generate the box on the right. The final result of this process is shown in Figure 4.

5. RESULTS AND EXAMPLES

The viewer we implemented was tested with the same six files used to review CityGML software in earlier sections of the paper. The files used show the features that the viewer is able to support as well a few limitations of the current implementation (akin to Table 2): File 1 – No, File 2 – Yes, File 3 – Yes, no ADE geometry, File 4 – No, File 5 – Yes, File 6 – Yes, only some properties.

The first file was used to test the viewer's support for large files. Unfortunately, it was unable to handle the size of the model and was unable to load the geometry. This was an issue we faced with other large files that we tried viewing as well. The second, third and fourth files contained ADE information in various forms and were used to test what kinds of cases the viewer was able to support. The second file, with the NoiseADE, only contained ADE information in the form of properties attached to native CityGML elements. The viewer was able to successfully show all the NoiseADE properties when the building is selected. The third and fourth files, however, contained ADE geometry as well. The viewer was able to load the native CityGML geometry and show the EnergyADE properties for the third file, with missing EnergyADE thermal surfaces and other ADE geometry. Meanwhile, for the fourth file, because all of the geometry are UtilityNetworkADE elements, nothing was able to be displayed. In fifth file, we were exploring different ways to represent location and had implemented a Local Engineering CRS to position the file instead of an EPSG code that was used in the other examples. The viewer was able to use this CRS to properly locate the building on the map. With the sixth file, we wanted to test if it was able to view CityGML 3.0 files, without any changes to the implementations. We used the existing code developed extract rules from the CityGML 2.0 schema to generate a new JSON file for the rules using a sample of the CityGML 3.0 XSD. Using this new JSON file, the viewer was able to properly display this sixth file as well. However, we realised that the way the various building elements were defined was different from earlier examples and thus the colour coding of the objects did not apply. We also realised that ADE elements were not able to be displayed properly, as they took the form of multiple elements with the same name.

6. DISCUSSION AND LIMITATIONS

Based on the observations and issues faced when trying to view different types of files, we identified some main limitations which sparked some ideas for future explorations and work.

6.1 Performance

One main limitation of the current demonstrative viewer is that it has poor memory performance and can only support models up to about 50MB in size. This is due to inefficient use of the entity API to create each polygon with its properties and keep the entire model in memory instead of using more efficient solutions like glTF models and 3D tiles that Cesium also supports. This choice was made to avoid the need to preprocess the CityGML files, as well as because of the lack of a simple and accessible method to convert CityGML to glTF object by object, rather than building by building, with common

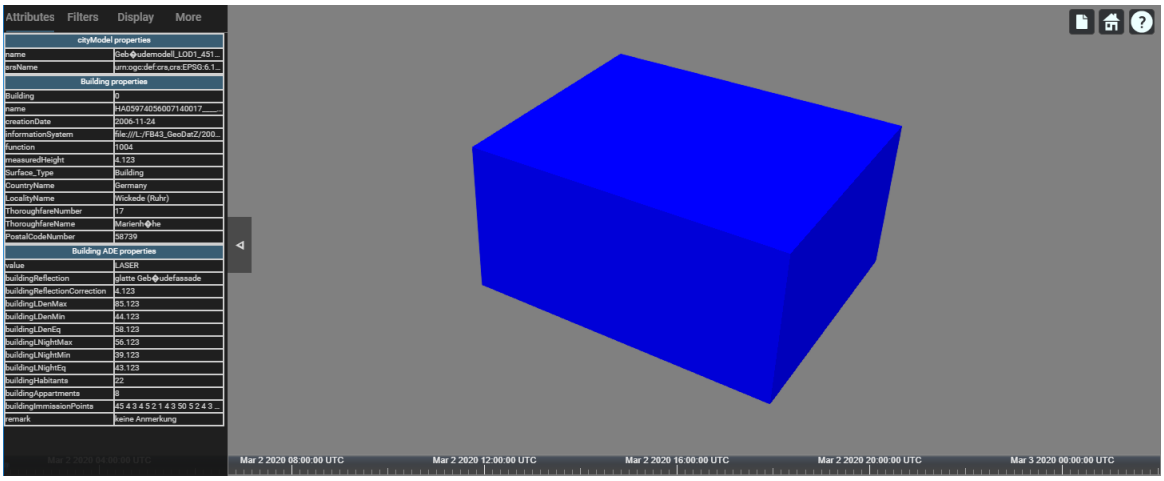

Figure 4. Result of the process described in Section 4.

conversion methods requiring an intermediary file format like COLLADA and using the COLLADA2glTF converter (Khronos Group, 2019). However, it would be beneficial to further investigate methods that make use of these file formats to improve performance for larger files in the future. Some interesting directions could be to try adapting 3DCityDB and its web client to accommodate building objects and generic rules. It would also be interesting to look deeper into conversion to glTF that maintains some semantic information as glTF also has the capacity to represent a model as a collection of scenes and nodes with additional custom properties attached, although these are typically used for animation purposes and may not be appropriate (Khronos Group, 2020).

6.2 Rule coverage

Another limitation is that many cases are not supported in our current implementation. Currently the rules derived from the XSDs are focused on viewing simple files containing elements from the Building module and support for other modules and more complex files are not fully supported. This is due to the simple rules used to process the XSD and CityGML files. In CityGML there are more complex relationships such as xlink references and implicit geometry that will need to be considered to fully support all aspects of CityGML. In addition, all the ADE information is only treated as simple text content. Geometry or other types of information are not properly considered at the moment. Only the final value of the ADE nodes are extracted as well, so not all information from the ADE nodes are fully considered and extracted. It may be beneficial to implement proper class binding for the native CityGML schema, such as in other CityGML software, and more effort will need to be spent defining rules and their conditions to ensure that all objects are treated properly. An approach that fully tracks and analyses the paths found in the file in order to determine which rules to use or which properties are assigned to which object might be one way to do this.

7. CONCLUSION

CityGML usage and visualisation is now integrated with a variety of applications and information sources and it has been adopted in a number of domains and locations. ADEs are also developing at a fast pace, with many being used today for a wide range of purposes. BIM to CityGML conversion now also has a range of support for mapping and simplifying geometry and translating properties and attributes that could aid a variety of use cases. However, it may still be difficult to make proper use of all these new developments, due to the fact that support and coordination between the separate elements are still developing. There are only a few platforms that are able to access and interact with all the newly obtained information and utilise them in a useful manner. Often, many layers of conversion, data structures and platforms still have to be implemented in order to utilise the data efficiently. In this paper we have developed a methodology towards bridging these gaps, and demonstrated it by implementing a web-based viewer supporting the visualisation of CityGML ADE-enabled files. We believe that our work, while preliminary, will contribute to increasing the usability of datasets with ADEs, which otherwise can hardly be visualised. Although we have picked up some features and developed a demonstration of a simple platform that would be able to view the information in CityGML models with ADE attributes at a building scale, much work still has to be done in order to come up with a useful platform that would serve the multitude of integration problems such models aim to tackle. For future work, we plan to improve the performance of the viewer, increase its functionality (e.g. support new types of geometries that can be defined by an ADE), and support CityJSON, a JSON-based implementation of CityGML, which recently has started supporting extensions akin to CityGML ADEs (Ledoux et al., 2019) and is gaining software support (Vitalis et al., 2020).

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

References

3D Geoinformation group, 2019. Open datasets created with 3dfier. https://3d.bk.tudelft.nl/opendata/3dfier/, retrieved on 22 Mar 2020.

Agugiaro, G., Benner, J., Cipriano, P. and Nouvel, R., 2018. The Energy Application Domain Extension for CityGML: enhancing interoperability for urban energy simulations. *Open Geospatial Data, Software and Standards* 3(1), pp. 139.

Amato, M., n.d. Cesium ion Brings 3D Tiling for Point Clouds, Photogrammetry, and 3D Buildings to the Cloud. https://cesium.com/blog/2018/10/09/ion-3d-tiles-pipeline/, retrieved on 22 Mar 2020.

Arroyo Ohori, K., 2020. azul: a fast and efficient 3D city model viewer for macOS. *Transactions in GIS*.

Atazadeh, B., Rajabifard, A. and Kalantari, M., 2017. Assessing Performance of Three BIM-Based Views of Buildings for Communication and Management of Vertically Stratified Legal Interests. *ISPRS International Journal of Geo-Information* 6(7), pp. 198.

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

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S. and Çöltekin, A., 2015. Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information* 4(4), pp. 2842–2889.

Boates, I., 2019. Sample of UtilityNetwork ADE data model using water network from Nanaimo, Canada. https://github.com/iboates/CityGML-UtilityNetwork-ADE-Nanaimo-Water-Network-Sample, retrieved on 22 Mar 2020.

Boeters, R., Arroyo Ohori, K., Biljecki, F. and Zlatanova, S., 2015. Automatically enhancing CityGML LOD2 models with a corresponding indoor geometry. *International Journal of Geographical Information Science* 29(12), pp. 2248–2268.

Chaturvedi, K. and Kolbe, T. H., 2016. Integrating dynamic data and sensors with semantic 3D city models in the context of smart cities. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* IV-2/W1, pp. 31–38.

Chen, L.-C., Wu, C.-H., Shen, T.-S. and Chou, C.-C., 2014. The application of geometric network models and building information models in geospatial environments for fire-fighting simulations. *Computers, Environment and Urban Systems* 45, pp. 1–12.

Czerwinski, A., Kolbe, T. H., Plümer, L. and Stöcker-Meier, E., 2006. Interoperability and accuracy requirements for EU environmental noise mapping. In: H. Kremers and V. Tikunov (eds), *International Conference on GIS and Sustainable Development (InterCarto – InterGIS 12)*, Berlin, Germany, pp. 182–194.

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

Deng, Y., Cheng, J. C. P. 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, pp. 1–21.

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

El-Mekawy, M., Östman, A. and Hijazi, I., 2012. An Evaluation of IFC-CityGML Unidirectional Conversion. *International Journal of Advanced Computer Science and Applications* 3(5), pp. 159–171.

Gesquière, G. and Manin, A., 2012. 3D Visualization of Urban Data Based on CityGML with WebGL. *International Journal of 3-D Information Modeling* 1(3), pp. 1–15.

Gröger, G. and Plümer, L., 2012. CityGML – Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing* 71, pp. 12–33.

Isikdag, U., Zlatanova, S. and Underwood, J., 2013. A BIM-Oriented Model for supporting indoor navigation requirements. *Computers, Environment and Urban Systems* 41, pp. 112–123.

Kang, T. W. and 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), pp. 1–11.

Khronos Group, 2019. COLLADA to glTF converter. https://github.com/KhronosGroup/COLLADA2GLTF/, retrieved on 22 Mar 2020.

Khronos Group, 2020. glTF Tools – Converters, Importers, and Exporters. https://github.com/KhronosGroup/glTF, retrieved on 22 Mar 2020.

KIT, 2017. KIT Sample files Energy ADE (Institute for Automation and Applied Computer Science (IAI) / Karlsruhe Institute of Technology (KIT)).

KIT, 2020. FZKViewer 5.3 (Build 1004) - Institute for Automation and Applied Informatics. https://www.iai.kit.edu/english/1302.php, retrieved on 22 Mar 2020.

Konde, A., Tauscher, H., Biljecki, F. and Crawford, J., 2018. Floor plans in CityGML. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* IV-4/W6, pp. 25–32.

Kumar, K., Ledoux, H. and Stoter, J., 2018. Compactly representing massive terrain models as TINs in CityGML. *Transactions in GIS* 22(5), pp. 1152–1178.

Kutzner, T., Chaturvedi, K. and Kolbe, T. H., 2020. CityGML 3.0: New Functions Open Up New Applications. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science* pp. 1–19.

Ledoux, H., Arroyo Ohori, K., Kumar, K., Dukai, B., Labetski, A. and Vitalis, S., 2019. CityJSON: A compact and easy-to-use encoding of the CityGML data model. *Open Geospatial Data, Software and Standards* 4, pp. 4.

Lim, J., Tauscher, H. and Biljecki, F., 2019. Graph transformation rules for IFC-to-CityGML attribute conversion. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* IV-4/W8, pp. 83–90.

Liu, X., Wang, X., Wright, G., Cheng, J., Li, X. and 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), pp. 53.

Noardo, F., Biljecki, F., Agugiaro, G., Arroyo Ohori, K., Ellul, C., Harrie, L. and Stoter, J., 2019. GeoBIM benchmark 2019: intermediate results. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XLII-4/W15, pp. 47–52.

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

Pispidikis, I. and Dimopoulou, E., 2016. Development of a 3D WebGIS system for retrieving and visualizing CityGML data based on their geometric and semantic characteristics by using free and open source technology. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* IV-2/W1, pp. 47–53.

Prandi, F., Devigili, F., Soave, M., Di Staso, U. and De Amicis, R., 2015. 3D web visualization of huge CityGML models. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XL-3/W3, pp. 601–605.

Prieto, I., Izkara, J. L. and del Hoyo, F. J. D., 2012. Efficient Visualization of the Geometric Information of CityGML: Application for the Documentation of Built Heritage. In: *Computational Science and Its Applications – ICCSA 2012*, Springer, Berlin, Heidelberg, Berlin, Heidelberg, pp. 529–544.

Simoes, B., Prandi, F. and Amicis, R. D., 2015. i-scope: A city GML framework for mobile devices. In: *2015 2nd ACM International Conference on Mobile Software Engineering and Systems*, IEEE.

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), pp. 355.

van den Brink, L., Stoter, J. and Zlatanova, S., 2013. UML-Based Approach to Developing a CityGML Application Domain Extension. *Transactions in GIS* 17(6), pp. 920–942.

Vitalis, S., Arroyo Ohori, K. and Stoter, J., 2020. CityJSON in QGIS: Development of an open-source plugin. *Transactions in GIS*.

Wang, K., Liu, G., Zhai, M., Wang, Z. and Zhou, C., 2018. Building an efficient storage model of spatial-temporal information based on HBase. *Journal of Spatial Science* 64(2), pp. 301–317.

Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubauer, A., Adolphi, T. and Kolbe, T. H., 2018. 3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML. *Open Geospatial Data, Software and Standards* 3(1), pp. 208.

Zhu, J., Wright, G., Wang, J. and 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), pp. 66–16.