CONCEPTUAL MODEL OF CITYGML RESTFUL WEB SERVICE

I. Pispidikis [1]*, E. Dimopoulou [1]

[1]School of Rural and Surveying Engineering, National Technical University of Athens, 9 Iroon Polytechneiou str, 15780 Zografou, Greece (pispidikisj@yahoo.gr; efi@survey.ntua.gr)

KEY WORDS: RESTful, Web Service, CityGML, 3D city model, Information Retrieval (IR)

ABSTRACT:

A 3D city model is considered as the digital representation of a city that may decompose into its objects (such as buildings, roads, railways, terrain, water, vegetation etc.) with clearly defined semantics, spatial and thematic properties. Depending on the level of detail, these objects may further decompose into more detailed features. The OGC standard CityGML, optimally allows integration of the diversified geoinformation of the aforementioned elements and provides multiple resolutions at different levels of detail (LoDs). Retrieving information from this standard is a challenge, in terms of interoperability, implementing the Web Service Technology. Within this technology, this paper examines and presents the conceptual model of the CityGML RESTful Web Service regarding all main resources which are structured according to the ten thematic modules of CityGML, in all available LoDs.

1. INTRODUCTION

3D city models are increasingly used in a wide range of applications beyond mere visualization. The optimal standard for the presentation of the said models is the CityGML due to the fact that it is a common semantic information model for the representation of 3D urban objects that can be utilized in different applications (Gröger, et al., 2012). Therefore, the interoperable and easy- to –use data retrieval of the CityGML model is vital to be achieved (Pispidikis and Dimopoulou, 2018). In this context, the implementation of Web Services technology provides a new approach in terms of system interoperability, while it overcomes the complexity of the need to convert data and install the appropriate programs (Fu and Su, 2010).

1.1 Related Work

Several OGC Web Services are currently available to retrieve and portray 3D geospatial data, such as W3DS (Web 3D Service), WVS (Web View Service) and WFS 2.0 (Web Feature Service 2.0). However, regarding the CityGML model, the WFS is mostly implemented, while it retrieves the real data and not the representation of the source (images or scene) (Pispidikis and Dimopoulou, 2018). However, serving CityGML via OGC WFS 2.0 presents a number of technical problems with regard to the structure of the CityGML models and the fact that CityGML schema is much more complex than those usually deployed in WFS. For that reason, the extension of the OGC WFS 2.0 has been further examined so that the complex structure of CityGML can be successfully retrieved (Curtis, 2008; Yao et al., 2018; Kolbe et al., 2009; Zhu, et al, 2016; Pispidikis and Dimopoulou, 2016). It should be noted that the WFS 2.0 and previous versions used a Remote-Procedure-Call-Over-HTTP architectural style implementing XML for any payload. This architecture was considered state-of-the-art when the WFS standard was originally designed in the late 1990s and early 2000s (Portele and Vretanos, 2018). However, the evolution of the Web 2.0 phenomenon has led to the increased adoption of the RESTful Service paradigm which takes full advantage of the web technology, making correct use of the HTTP protocol and also follows the ROA (Resource-Oriented Architecture) (Pispidikis and Dimopoulou, 2018). This REST-based architecture was adopted by the upcoming WFS version 3 (Portele and Vretanos, 2018), now called OGC API-Features. Therefore, this version of the WFS standard uses a resource architecture and specifies a RESTful service interface providing resources regarding features and feature collection respectively. So, the list of a feature collection (e.g. buildings) can be retrieved using the following request:

../collections/buildings/items

Thereafter, each feature in a feature collection can also be requested by implementing the respective id as sub-resource.

../collections/buildings/items/{id}

The REST-based architecture was also adopted by the CityGML RESTful Web Service (Pispidikis and Dimopoulou, 2018) which focuses on retrieving CityGML data by taking into account the semantic aspect of 3D city models rather the geometric. So, with regard to the CityGML RESTful Web Service, several principles and guidelines were addressed and the conceptual design of the resource and child resources regarding the building module of CityGML was presented (Pispidikis and Dimopoulou, 2018). Also, Athanasiou et al. (2018) upgraded the 3D WebGIS application as developed and presented by Pispidikis and Dimopoulou (2016), by integrating the “bldg” resource of CityGML RESTful Web Service for the assessment of the proposed methodology in which they semantically built a building model in BIM and CityGML format.

The aim of this paper is to conclude the conceptual model of the CityGML RESTful Web Service by conceptual designing the rest main resources at all semantics LoDs. According to the architecture of the CityGML which is shaped via five key components, only the component of the ten thematic modules defines the semantic features of the basic objects of a 3D city models. Hence, these thematic modules were used as the main resources of the CityGML RESTful Web Service (Figure 2). Additionally, the semantic enrichment of the said features is either based on LoDs or not. Therefore, some of these resources have LoD- based sub-resources and hence, their semantic

* Corresponding author

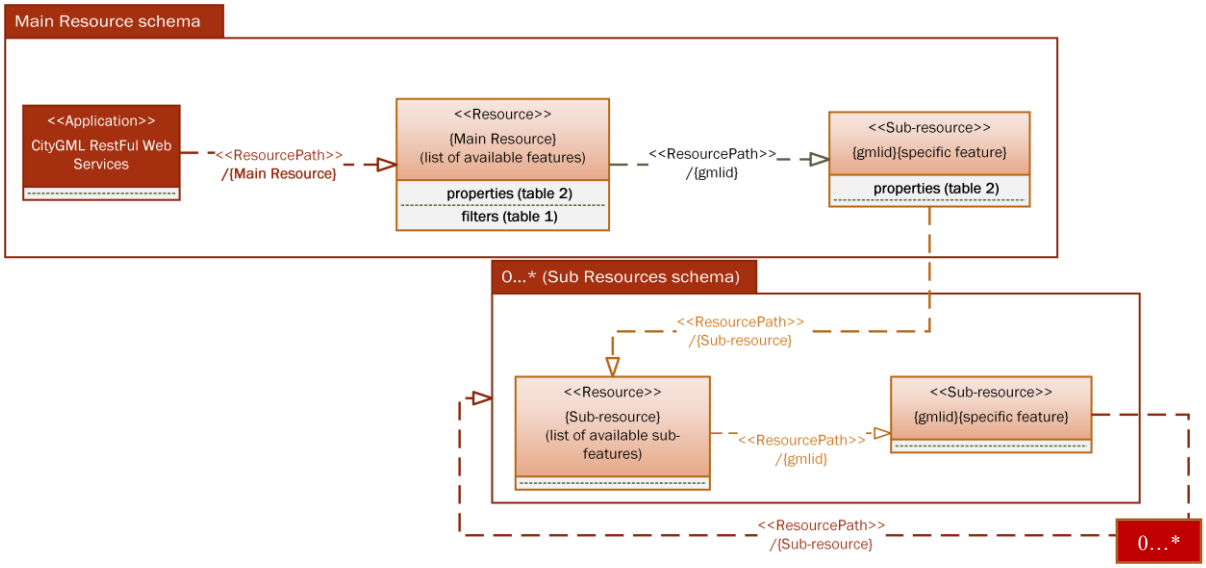

Figure 1. Retrieval resource schema of CityGML RESTful Web Service

retrieval is available based on the LoD, while, some resources are LoD- independent and so there is no differentiation regarding their semantics sub-resources from one LoD to another.

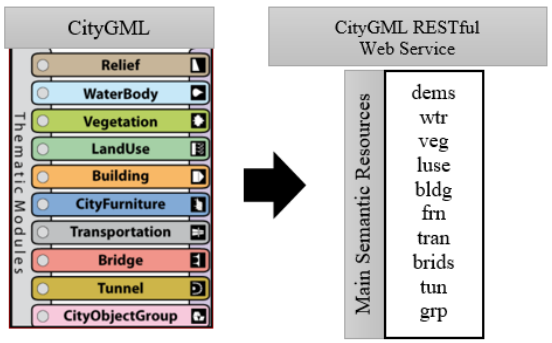

Figure 2. Main Resources of CityGML RESTful Web Service

Section 2 presents the information retrieval and filters of the available resources. Also, the retrieval resource schema of CityGML RESTful Web Service is described. In section 3 the main resources are briefly described. Next, in section 4 LoD-independent sub-resources are presented, while in section 5 the LoD-based sub-resources are described. Finally, section 6 concludes the findings and discusses solutions and suggestions regarding the compatibility of the CityGML RESTful Web Service with the upcoming CityGML version 3 and also the multiple techniques that can be implemented, so that the Cross-Domain issues can be overcome.

2. INFORMATION RETRIEVAL AND FILTERING

The names of the available main resources of the CityGML RESTful Web Service are based on the namespace prefix of CityGML specification (Figure 2) and the response of each request is mainly a list of the available thematic modules respectively. Each thematic module of this list contains general information according to CityGML specification. Moreover, most of the main resources can be filtered using the general filters such as function, usage, class, bbox and lod (table 1) (Pispidikis and Dimopoulou, 2018).

Filter	URI (Example by using brids resource)
function	../ brids?function=3020
usage	../ brids?usage=1010
class	../ brids?class =1000
bbox	../ brids?bbox= 334433.0,4455667.0,445677.0,5566556.0
lod	../ brids?lod=3

Table 1. General filters

It should be noted that the information retrieval about a specific main resource can be achieved implementing the respective gmlid as sub-resource. Furthermore, the general value types of the retrieval properties regarding main resources are described in table 2.

Information	Type	Description
lod	Number	LoD value
XXX*Part	Boolean	True or False
isMovable	Boolean	True or False
XXX*Infomation	Object	List of key value pairs based on respective module
geometry	Object	Geometry object based on GeoJSON specification or external format
generic	Object	Ad hoc list of key value pairs based on generic module
address	Object	List of key value pairs based on xAL specification
links	Object	List of key value pairs regarding links to the parent and child resources
gmlid	String	Gmlid value

*The characters "XXX" are defined based on the name of the main resource (e.g bridsPart, bridsInformation)

Table 2. Available information of the main resources

According to the Richardson maturity model (Fowler, 2010), the "links" property is considered very important to be implemented so that the implementation of HATEOAS (Hypermedia As the Engine of Application State) is achieved and hence the CityGML Web Service to be RESTful. As a result, each resource must contain information regarding links to other available resources. Consequently, the available links of a resource of the CityGML RESTful Web Service contain links to all relative resources (itself, parents and child).

The generic information retrieval schema regarding the main resources and the respective sub-resources of the CityGML RESTful Web Service is schematically shown in Figure 1. This schema consists of two sub-schemas. The first one describes the retrieval mechanism of the main resources. The name of the main resource is defined according to the main semantic resources (see Figure 2). So, the list of the main resources regarding the specific module can be retrieved using the following request:

../{main resources}

Thereafter, a specific main resource can also be retrieved implementing the gmlid property of the desirable module.

../{main resources}/{gmlid}

The second sub-schema describes the retrieval mechanism of the respective sub-resources. Each main schema could contain 0 to many sub-schemas and each sub-schema could also contain 0 to many sub-schemas. The sub-resources are defined according to the semantic enrichment of the thematic modules of CityGML and further described and presented in section 4 and 5.

3. MAIN RESOURCES

Generally, the main resources of CityGML RESTful Web service are defined according to the thematic modules of CityGML (Figure 2). However, some extra main semantic resources were also designed to make it easier to access their available semantic features. These extra main resources are part of the main resources such as "tran" and "veg" (Figure 3).

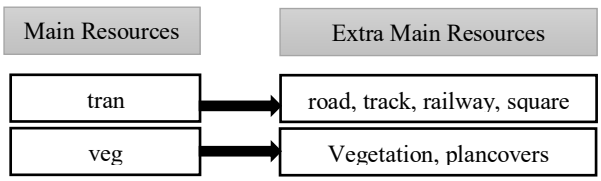

Figure 3. Extra main resources

3.1 Brids Resource

The "brids" resource is the main thematic resource regarding the bridge module of the CityGML. This resource retrieves a list of the available bridges and bridge parts respectively. This list can be limited using not only the aforementioned general filter (see table 1) but also the "bridsPart" and the "isMovable" filter parameters. The value of these filters is Boolean and provide information about whether the bridge is bridge part or not and whether is movable or immovable respectively.

3.2 Tun Resource

The "tun" resource refers to the tunnel module of the CityGML. The scope of the tunnel model encompasses manmade structures that are located mostly below the terrain surface and are intended to convey transportation flows such as pedestrians, car, trains, etc. Hence, the geological structures, natural caves, mining facilities, and subsurface utility network are excluded (Gröger, et al., 2012). The available filters for this resource are the general filters (see table 1) and also the tunPart (see table 2). The value type of the latter is Boolean and describes whether the tunnel is tunnel part or not.

3.3 Tran Resource

The transportation module of CityGML can be requested by implementing the "tran" main resource. The information retrieval is the available transportation models grouped by the predefined subclasses of CityGML such as road, track, railway and square (Figure 4). Thereafter, this resource can be filtered using a new filter parameter called "categories" with values the respective subclasses of CityGML. It should be noted that multi category values can be used simultaneously separating them with a comma. Thereafter, in each group category, the corresponding resource link of the specific transportation model can also be retrieved.

../tran?categories= road,square

The four predefined subclasses are extra main resources and not sub-resources, since they are independent of the "tran" resource (see Figure 3). It should be noted that for both LoD0 and LoD1 there are no extra semantic characteristics for these resources. Hence, the child resource of said resources regarding the aforementioned LoDs is only the gmlid value.

../roads/{gmlid}

```
{
  "type": "object",
  "properties": {
    "category": {"type": "string"},
    "counts": {"type": "number"},
    "links": {
      "type": "object",
      "properties": {
        "link": { "type": "string"},
        "rel": {"type": "string"}
      }
    }
  }
}
```

Figure 4. "tran" resource in JSON schema

3.4 Wtr Resource

The retrieval of the Waterbody module of the CityGML can be achieved using the "wtr" resource. The information retrieval of this URI (Uniform Resource Identifier) is a list of waterbody models and each of these models contains the following properties: lod, wtrInformation, geometry, generic, gmlid and links. Also, the general filters (see table 1) can be implemented when the "wtr" resource is requested. Additionally, for both LoD0 and LoD1, the only sub-resource is the respective gmlid value.

3.5 Veg Resource

The vegetation objects can be requested using the "veg" resource. These objects of CityGML distinguish into solitary vegetation objects like trees and vegetation areas, which represent biotopes like forest or other plant communities. Hence, the information retrieval of the "veg" resource is the available vegetation objects grouped on the basis of the aforementioned categories such as vegetation (solitary vegetation objects) and plantcovers (vegetation areas). Next, this resource can be filtered using a new filter parameter called "vegetationtype" with values according to the above categories. Then, in each group, the corresponding resource link of the specific vegetation model will also be retrieved including the respective available information such as vegInformation, generic, gmlid, lod and links (see table 2). Additionally, the general filters (see table 1) can be used in both categories, while the vegetation category can also be filtered using the "species" parameter. The value of this parameter is defined according to the code list of CityGML specification regarding the solitaryVegetationObject attribute "species" (Gröger, et al., 2012). The "veg" resource is mainly used in order to inform the users about the two available group resources regarding the solitary vegetation objects and the vegetation areas. So, the URI resources of these categories are "vegetation" and "plancovers" respectively. These resources are extra main resources, since they are resource-independent from the "veg" resource (see Figure 3). Thereafter, the retrieval of specific object is achieved by implementing the respective gmlid as sub-resource.

../vegetation/{gmlid}

3.6 Luse Resource

The "luse" resource retrieves information with regard to the LandUse model of the CityGML. The LandUse model can be used to describe areas of the earth's surface dedicated to a specific land use, but also to describe areas of the earth's surface with specific land cover, with or without vegetation, such as sand, rock, mud flat, forest, etc. Furthermore, this model represents both the land use and the land cover concepts. The first describes the human activities on the earth's surface whereas the second one describes its physical and biological cover. Hence, the information retrieval of the "luse" resource is a list of the aforementioned concepts which include the following information: luseInformation, lod, gmlid, links, generic and geometry (see table 2). Moreover, the implementation of the general filters (see table 1) is available. The "luse" resource are simple URI with no extra sub-resource except for the gmlid, since there is no extra semantic enrichment according to the LandUse module of CityGML.

3.7 Frn Resource

The "frn" resource refers to the city furniture module of the CityGML. The objects of this module are immovable objects like lanterns, traffic lights, traffic signs, or bus shops and can be found in traffic areas, residential areas, on squares or in built-up areas. The response by executing the "frn" resource contains the following key-value properties: frnInformation, lod, gmlid, links, generic and geometry (see table 2). Also, the general filters (see table 1) can be utilized. Similar to the "luse" resource, the "frn" resource is also a simple URI.

3.8 Dem Resource

An essential part of a city model is the terrain and therefore, the DTM (Digital Terrain Model) is provided by the relief module of CityGML. The information about this module can be retrieved by implementing the "dem" resource and contains the following properties: lod, links, generic and gmlid. Additionally, this resource can be filtered by using only the lod filter parameter.

3.9 Grp Resource

The aggregation of the requested CityObjects can be achieved by utilizing the CityObjectGroup module of CityGML. Thus, the "grp" main resource can be implemented for the retrieval of the said module. Thereafter, apart from the general information (gmlid, grpInformation and generic), a new property called "group" is defined. This property is an object that contains a list of the grouped CityObjects together with their respective links. Moreover, this resource can be filtered by using the function, usage and class filter parameters.

4. LOD-INDEPENDENT SUB-RESOURCES

The thematic modules of CityGML allow the representation of the thematic and spatial parameters of the 3D models' objects at different levels of detail. The transition from one level to another imposes and allows different semantic details both on the outside and inside. Consequently, the sub-resources of the main resources should be designed based on LoDs. However, the majority of the thematic modules of CityGML are enriched with semantic characteristics either independently of LoDs or from LoD2 and above without any differentiation from one level to another. Hence, the sub-resources of these main resources will be available for all LoDs or from LoD2 and above.

4.1 Sub-Resources for all LoDs

The main resources that their semantic features are independent of LoDs are the "grp", "dem", "frn", "luse", "veg", "vegetation" and "plancovers" resources. Thus, these resources can be implemented for all available LoDs of CityGML. However, apart from the "dem" resource, the rest of the said resources have no extra sub-resources except for gmlid. So, the "dem" resource is used to retrieve a list of available reliefs. Thereafter, when a specific relief is requested by implementing the respective gmlid as sub-resource, then four sub-resources are available such as "tins", "masspoints", "breaklines" and "raster". These sub-resources are defined according to the CityGML specification for the relief module, since each relief may consist of one or more of these entities. For example, the retrieval of all available tins of specific relief is achieved by using the following request:

../dem/{gmlid}/tins

4.2 Sub-Resources from LoD2 and Above

The "tran" and "wtr" resources belong to the case that their sub-resources are only available from LoD2 and above, without any differentiation.

4.2.1 Tran Sub-Resource: From LoD2 and above the four resources (road, track, railway and square)(see 3.3) are further subdivided semantically into TrafficAreas, which are used by transportation, such as cars, trains, public transport, airplanes,

bicycles or pedestrian and in AuxilliaryTrafficAreas, which are of minor importance for transportation purposes. The URIs of these child resources are "auxiliaries" and "trafficareas" respectively. For example, the retrieval of specific traffic area from a specific road is achieved by using the following request:

../roads/{gmlid}/trafficareas/{gmlid}

4.2.2 Wtr Sub-Resource: In LoD2 and higher LoDs, the water body is bounded by a distinct semantic surface such as WaterSurface, which is defined as the boundary between water and air, WaterGroundSurface, which is defined as the boundary between water and underground and WaterClosureSurface, which is the virtual boundary between water bodies or between water and the end of a modelled region. As a result, the above features are the child resources of "wtr" resource using the following URIs respectively: "water", "grounds" and "closures". It should be noted that the "water" sub-resource has not the gmlid as child resource because according to the CityGML WaterBody module, a waterbody must have one or zero WaterSurface. So, for retrieving the WaterSurface of specific WaterBody the following request should be implemented:

../wtr/{gmlid}/water

5. LOD-BASED SUB-RESOURCES

The bridge and tunnel modules of CityGML belong to the category of modules where their semantic details are enriched on transition from one level to another. Therefore, the availability of the sub-resources of the "brids" and "tun" resources are based on the LoD.

5.1 Brids Sub-Resource

Firstly, the semantical richness of bridge module increases from LoD1 and above and therefore, the child resource of "brids" resource regarding LoD0 is only the gmlid value. Thereafter, in LoD1 the semantic elements named BridgeContructionElement are included. These features are considered essential from a structural point of view like pylons, anchorages etc. Thus, the "construction" URI is defined as child resource. The information retrieval of said resource is a list of the available BridgeConstructionElements. This list can be filtered using the general filters (see Table 1). Additionally, each of these elements can be retrieved using the corresponding gmilid as sub-resource.

5.1.1 LoD2 brids sub-resources: Except for the BridgeConstructionElement, the additional supported semantic characteristics of the LoD2 Bridge are the exterior boundary surfaces (WallSurface, RoofSurface, GroundSurface, OuterFloorSurface, OuterCeilingSurface), the ClosureSurface, and the BridgeInstallation. Consequently, these semantic features are the LoD2 child resources of the "brids" resource. The URIs regarding boundary surfaces are "walls", "roofs", "grounds", "floors" and "ceilings" respectively and retrieve a list of the corresponding thematic surfaces. Moreover, with regard to the BridgeInstallation, the "installations" child resource is defined. This resource can be filtered using several filters such as usage, function, class and type. Additionally, a new property "type" is defined for the separation of the interior and exterior bridge installations, which is enabled only for the LoD4. So, the value of this property is either interior or exterior. Finally, the "closures" resource is embedded so that the open sides of bridge can be virtually closed by using the ClosureSurface. It should be noted that all of the above-mentioned sub-resources have as child resource the respective gmlid value and hence any specific semantic feature can be requested. An instance of a specific wall request is the following:

../brids/ {gmlid}/walls/{gmlid}

5.1.2 LoD3 brids sub-resources: The additional semantic features of the LoD3 bridge module are the opening features (windows and doors). Consequently, the respective resources of the aforesaid features are considered vital to be defined. Hence, the URI of these resources are "windows" and "doors". These resources are child resource of each LoD3 "walls" and "roofs" sub resources with regard to "brids" resource. The retrieval of specific data regarding the aforementioned resources is achieved implementing the corresponding gmlid as sub-resource. Additionally, each specific "doors" resource should contain information regarding the address and, therefore, the object address is defined with allowable values in compliance with the xAL specification.

5.1.3 LoD4 brids sub-resources: Firstly, the property "type" of the sub-resource "installations" is enabled so that the separation of the interior and exterior installations is achieved. Moreover, the "rooms" child resource is defined and the respective list of the available rooms of a bridge can be retrieved. Thereafter, each room provides several links for child resources such as "furniture", "installations", "walls", "floors", "ceilings" and "closures". The first one retrieves a list of furniture that are located in a specific room. The accessible information of this resource is class, usage, function, gmlid, generic, appearance, geometry and links. Additionally, the available filter parameters of the "furnitures" resource are class, usage and function. In this context, the rest child resources (installations, walls, floors and ceilings) retrieve a list of the respective available semantic features. Moreover, the "closures" child resource is also embedded so that the opening space that is not filled by a door or window can be sealed by a virtual surface called ClosureSurface. Generally, the retrieval of a specific semantic feature is achieved using the respective gmlid. It should be noted that in the LoD4 there are two sub resources with the same name but different URIs. The name of these resources is named "installations". The first one is child resource of "brids" resource and retrieve a list of interior installations in a specific bridge, while the second one is the child resource of the "rooms" resource and retrieve the respective installations that are located in a specific room.
Similar to the LoD3, the interior boundary resources (walls and floors) provide the "windows" and "doors" child resources. The aforesaid resources have similar properties and filters like LoD3 opening resources.

5.2 Tun Sub-Resource

The tunnel model is closely related to the building and bridge models of CityGML. However, this module supports the representation of semantic aspects of tunnels and tunnel parts only in four levels of detail, LoD1 to LoD4. With regard to LoD1, there is only the gmlid value as sub-resource since no extra semantic features are defined according to the CityGML specification. In LoD2 and higher LoDs the outer structure of a tunnel can also be differentiated semantically and therefore, additionally child-resources are designed based on the LoD.

5.2.1 LoD2 tun sub-resources: The LoD2 child resources of the "tun" resource are based on the respective classes' _BoundarySurface (exterior) and TunnelInstallation of tunnel module of CityGML. As a result, in terms of the outer boundary

surface, the following URIs are specified: "walls", "grounds", "roofs" and "ceilings". These resources retrieve the semantics structure regarding the exterior shell of a tunnel as well as the exterior visible surface of the HollowSpace (semantic object for modelling the free space inside a tunnel regarding LoD4). Additionally, the "closure" resource is also defined so that the open side of the model that was sealed by virtual surface can also be retrieved. The semantic objects which refer to the outer component of a tunnel and strongly affect its outer characteristics belong to TunnelInstallation feature. So, the "installations" resource is used for the retrieval of the aforementioned objects. This resource can be filtered by implementing the following filters: class, function, usage and type. The first three filters are defined in compliance with the respective attributes of the feature TunnelInstallation of CityGML. It should be noted that the property "type" is embedded to the "installations" resource and thus the separation of the interior and exterior installation is achieved. This type is also used as filter.

5.2.2 LoD3 tun sub-resources: From LoD3 and above the exterior boundary surface may contain opening features like doors and windows. Therefore, for each wall or roof sub-resource, the list of the available openings can be retrieved by using the "doors" and "windows" child resources respectively. An instance of a specific door request is the following:

../tun/ {gmlid}/walls/{gmlid}/doors/{gmilid}

5.2.3 LoD4 tun sub-resources: The free space inside a tunnel or tunnel part is semantically described by the HollowSpace. Therefore, the "hollowspaces" child resource regarding a specific tun resource can be implemented and thus the respective list of the HollowSpaces, including their available information (class, usage, function, gmlid, generic, appearance, geometry and links), can be retrieved. Each hollow space can be semantically described and modelled by specialized boundary surfaces, such as FloorSurface, CeilingSurface, InteriorWallSurface and ClosureSurface. Thereafter, each "hollowspace" resource provides several links to the respective boundary child resources (walls, floors, ceilings and closures). Then, a specific "walls" resource may provide as child resources the opening feature such as windows and doors. Moreover, the objects inside a tunnel which are permanently attached to the tunnel structure and cannot be moved can be requested by using the "installations" resource. It should be noted that there are two available "installations" sub-resources with different URIs so that the interior installation can be retrieved based on either a specific tunnel or a specific hollow space. An instance of the above-mentioned cases is the following:

../tun/ {gmlid}/hollowspaces/{gmlid}/installations

../tun/ {gmlid}/installations?type=interior

Additionally, the request of the movable objects of a hollow space can be achieved by implementing the "furnitures" sub resource.

6. DISCUSSION AND CONCLUSIONS

A 3D city model is considered as the digital representation of a city that may decompose into its objects with clearly defined semantics, spatial and thematic properties. The OGC standard CityGML, optimally allows integration of the diversified geoinformation of the aforementioned elements and provides multiple resolutions at different levels of detail (LoDs). Therefore, the interoperable and easy- to –use mechanism for the data retrieval of a 3D city model should take into account the technology of web services as well as the evolution of the CityGML standard. Additionally, the semantic aspects of CityGML data should be also used as basic filter for data retrieval. In the context of this paper, the CityGML RESTful Web Service is proposed as suitable mechanism that meets the above-mentioned requirements (Pispidikis and Dimopoulou, 2018).
The utilization of the aforementioned REST-based service for CityGML 2.0 facilitates users to retrieve and manage 3D city models data without the need for knowledge of the complex structure of CityGML. Also, the resources and sub-resources of this service are based on the ten thematic modules of CityGML 2.0, and their availability depends on the LoDs. So, the sequential retrieval of the semantic features of CityGML is achieved. Additionally, through RESTful implementation, the CityGML RESTful Web Service follows several constrains such as addressability, uniform interface, statelessness, self-describing message and HATEOAS (Webber, 2010). Therefore, the service interact by exchanging request and response messages, which contain both the representations of resources and the corresponding metadata. Moreover, the URI of every next request can be retrieved from the "links" object of the current request. As a result, easy-to-use data retrieval can be completed by non-expert users.

The REST-based architecture was also adopted by the upcoming OGC API-Features leaving the Remote-Procedure-Call-Over-HTTP architectural style which is used by previous versions of WFS. The adoption of this architecture style utilizes the WOA (Web-Oriented Architecture) and hence, the development of reliable, flexible application is facilitated in a most easiest and economical way (Athanasiou et al., 2018). The OGC API-Features provides basic resources for retrieving features and feature collections. These resources are similar to the main resource schema of CityGML RESTful Web Service (see Figure 1). However, the core of OGC API-Features does not currently support the implementation of extra sub-resources but provides solution for this limitation by extending the Core API by including richer queries from existing OGC standards (Portele, 2019). The said solution is quite sufficient, since the OGC API-Features is intended to provide a general solution for retrieving data regarding all available standards of the OGC API family. However, this implies and requires good knowledge for both the structure of the source (e.g. CityGML) and the respective syntax of the implemented OGC standard such as OGC Filter Encoding Standard 2.0, OGC Common Query Language (CQL) or other query languages or data platforms such as Falcor and GraphQL (Portele, 2019).

The integration of the sub-resources schema of CityGML RESTful Web Service as an extension to the OGC API-Features will provide a sufficient way to semantically retrieve complex CityGML data. However, the aforementioned approach is out of the scope of the OGC API-Features, since the latter is not intended to implement just a standalone API but the same Web API should also implement other standards of the OGC API family (Portele, 2019).

CityGML RESTful Web Service was conceptually designed to be an information-based retrieval model regarding CityGML 2.0. Therefore, it is not geometrically-based like other OGC standards such as WFS 2.0 and OGC API-Features and thus, the retrieval format is a JSON schema with a list of information (see Table 2). One of this information could be the geometry object which can be retrieved in GeoJSON format. The JSON format facilitates the easy-to-use parsing and filtering of the retrieval data by client-

side programming languages such as JavaScript. Thereafter, this data can be further used as input parameters (descriptive or geometric) in spatial analysis tools.

6.1 Cross-Domain Issues

The execution of each request of CityGML RESTful Web Service is implemented in accordance with HTTP specification. Hence, for the retrieval of the data the HTTP GET method is implemented (Pispidikis and Dimopoulou, 2018). The utilization of this mechanism from distributed resources with different domains may have Cross-Domain issues. These issues mean that certain Cross-Domain requests will be forbidden by default by the same-origin security policy (W3C, 2010). Fortunately, the modern browsers support several techniques for overcoming these issues such as CORS (Cross-Origin Resource Sharing) and JSONP (JSON with Padding). The CORS is considered a standard and a mechanism that allows JavaScript on a web page to consume REST API served from a different origin. The CORS can be implemented through the HTTP Header "Access-Control-Allow-Origin", which can be enabled in a RESTful Web Service. An alternative way to share the data bypassing the same-origin policy without need of modern browser is the JSONP format which does not use the XMLHttpRequest object. Instead, it dynamically inserts <script> tag into a webpage that is not considered a Cross-Domain issue. However, apart from the aforementioned solutions, a proxy server can be utilized when executing the desirable request, avoiding all issues regarding the Same-Origin Policy. More specific, a proxy server can receive any request from distributed resource and then acting as a client on behalf of the user, requests the data from the server. Thereafter, when the data is returned, the proxy server relates it to the origin request and forwards it to the user.

6.2 Compatibility with CityGML 3.0

Since January 2018, a version of the CityGML 3.0 conceptual model has been made available in development mode on the original GitHub repository for OGC CityGML 3.0 (GitHub, 2018). This upcoming version brings a number of improvements, extensions and new functionalities; they are briefly presented in the following paragraphs, with an overview of the said changes and the degree of their impact on the conceptual model of CityGML RESTful Web Service.

6.2.1 New concept of LoD: In CityGML 3.0 the LoD concept is modified, based on the proposed LoDs as described by Lowner, et al. (2016). According to the authors, the main barrier in the current concept of LoD is that the interior structure of an element can only be represented if the exterior shell is represented in LoD4, which implies the highest semantic complexity and geometric detail. Therefore, in CityGML 3.0, LoD4 is replaced by LoD0 to LoD3 for exterior and indoor objects and all feature types can be represented in each LoD. So, it is possible to model the outside shell of a model in LoD1 while representing the interior structure in LoD2 or LoD3. It should be noted that the main structure of the CityGML RESTful Web Service is not affected by this important change as its conceptual model of the resources is designed by taking into account the semantic aspect of CityGML which remains the same. However, the availability of the sub-resources should be modified so that these resources can be provided at all LoDs (LoD0 to LoD3).

6.2.2 New Core Model: All spatial representations are rephrased based on the two pivotal abstract classes Space and SpaceBoundary. Therefore, the feature classes in the thematic modules represent their spatial characteristics almost exclusively using the aforementioned classes and no longer have direct associations with geometry classes. Nevertheless, the new Core model will not affect the conceptual design of the basic structure of the CityGML RESTful Web Service, since all available resources of this service are based on the thematic modules of the CityGML and not the spatial representation of these modules.

6.2.3 New modules: The CityGML version 3.0 will support the following three new modules: Construction, Versioning and Dynamizer. The Construction module groups all classes which are similar over different types of constructions like buildings, tunnels, bridges and introduces a new class "OtherConstruction" to represent other constructions not belonging to any of the aforementioned three modules. More specific, the construction elements refer to the boundary and opening surfaces regarding the modules building, bridge and tunnel which remain the same even if they belong to the construction module. So, the respective boundary and opening child resources of the CityGML RESTful Web Service will not be changed.

The Versioning module introduces bitemporal timestamps for all objects. Therefore, except of the attributes "creationDate and "terminalDate" from CityGML 2.0, all objects now can have a second lifespan expressed by the attributes "validFrom" and "validTo". Additionally, each geographic feature will have two identifiers: the "identifies" property and the "gml:id" attribute. The value of the "identifier" property will be stable along the lifetime of the real-word object, while the "gml:id" attribute will be constructed from the "identifier" with concatenated timestamp (Chaturvedi, et al., 2017). The versioning module can be supported by the CityGML RESTful Web Service by defining a new object "versioning" as an information retrieval for each resource. Additionally, this object should be included in the general filters. The object "versioning" will be a List of key value pairs based on the Versioning module of the CityGML 3.0.
The Dynamizer module improves the usability of CityGML for different kinds of simulations and also facilitates the integration of sensors with 3D city models. Through the Dynamizers, the link of timeseries data (OGC TimeseriesML, OGC observation and Measurement, tabulated data in external files like CSV) to a specific attribute or property of a specific object within the 3D city model is achieved (Chaturvedi and Kolbe, 2017). This capability facilitates the dynamic or real time updating of the source data and hence, it does not affect the mechanism of the retrieval of these data through CityGML RESTful Web Service.

6.2.4 The revised Transportation module: In the new Transportation module, the transportation objects such as road, track, railway and square, can be subdivided into sections. These sections can be regular road, track or railway legs, intersection areas or roundabouts, each belonging to multiple Road or Track objects. Thereafter, in order to avoid a redundant representation of this shared object, Xlinks will be used in the CityGML 3.0 instance document to reference the shared section (Beil, et al., 2017). Additionally, in LoD3, the representation of subtle structure such as manholes or roadway damages will be allowed. These holes in a street surface can extend over more than on road section and therefore will be modelled as a separate class. Moreover, TrafficArea and AuxiliaryTrafficArea will be changed to TraficSpace and AuxiliaryTrafficSpace. As a result, taking into account all the above mentioned changes, the sub-resources of the "tun" resource should be modified. Specifically, the URI "trafficareas" should be replaced by the URI "trafficspace", while three new resources named "drains", "manholes" and "roadwaydamages" regarding the LoD3 should be added. These resources will contain the following information: trnInformation, lod, generic, gmlid and geometry

(see table 2). Moreover, the said resources can be filtered by implementing the general filters (see Table 1). An instance of request for the list of roadway damages in a specific area is the following:

../roadwaydamages?bbox=334433.0,4455667.0,445677.0,5566556.0

6.2.5 New components of Building Module: The new Building module mainly remains the same. However, two new subdivision will be included, the "BuildingUnit" and the "Storey". These subdivisions will have Xlinks to the respective rooms. So, two new child resources of "bldg" resource should be embedded such as "buildingunits" and "storeys". These resources will retrieve a list of the respective building units and storeys. Thereafter, the retrieval of specific object is achieved by implementing the respective gmlid as sub-resource. This sub-resource will contain a list of links to the corresponding "rooms" sub-resources that includes. An instance of the implementation of the new sub-resources is the following:

../bldg/{gmlid}/buildingunits/{gmlid}

With the aforementioned request a specific building unit is retrieved and then the relative rooms can be retrieved as well.

ACKNOWLEDGEMENTS

The Onassis Foundation for the Scholarship of this research is gratefully acknowledged.

REFERENCES

Athanasiou, K., Pispidikis, I., Dimopoulou, E., 2018. Semantic-based Technologies for Interoperable BIM and GIS 3D Modelling, Storage and Retrieval. FIG Commission 3 Annual Meeting and Workshop 2018, 3-6 December, Italy: Naples

Beil, C. and Kolbe, T. H., 2017. CityGML and the streets of New York - a proposal for detailed street space modelling, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., IV-4/W5, 9-16, https://doi.org/10.5194/isprs-annals-IV-4-W5-9-2017

Chaturvedi, K., Kolbe, T H., 2017. Future City Pilot 1 Engineering Report. Retrieved from http://docs.opengeospatial.org/per/16-098.html (April 2019).

Chaturvedi, K., Smyth, C. S., Gesquière, G., Kutzner, T., Kolbe, T. H., 2017. Managing versions and history within semantic 3D city models for the next generation of CityGML. In Advances in 3D Geoinformation (pp. 191-206). Springer, Cham.

Curtis, E., 2008. Serving CityGML via web feature services in the OGC web services-phase 4 testbeds. Journal of Advances in 3D geoinformation systems, 331-340

Fowler, M., 2010. Richardson Maturity Model: steps toward the glory of REST. Retrieved from http://martinfowler. com/articles/richardsonMaturityModel. Html (April 2019).

Fu, P., Sun, J., 2010. Web GIS: principles and applications. USA: Esri Press.

GitHub., 2018. CityGML-3.0. Retrieved from https://github.com/opengeospatial/CityGML-3.0CM (April 2019).

Gröger, G., Kolbe, T., Nagel, C., Hafele, K.-H., 2012. OGC City Geography Markup Language (CItyGML) Encoding Standard. Retrieved from Open Geospatial Consortium Inc.: www.opengis.net/spec/citygml/2.0 (April 2019).

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

Kolbe, T. H., König, G., Nagel, C., Stadler, A., 2009. 3D-Geo-Database for CityGML. Institute for Geodesy and Geoinformation Science, Technische Universität Berlin, 2(1).

Löwner, M.-O., Gröger, G., Benner, J., Biljecki, F., and Nagel, C., 2016. Proposal for a new lod and multi-representation concept for CityGML, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., IV-2/W1, 3-12, https://doi.org/10.5194/isprs-annals-IV-2-W1-3-2016.

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, 47-53, https://doi.org/10.5194/isprs-annals-IV-2-W1-47-2016.

Pispidikis, I. and Dimopoulou, E., 2018. CityGML RESTful Web Service: automatic retrieval of CityGML data based on their semantics. Principles, guidelines and bldg conceptual design, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., IV-4/W6, 49-56, https://doi.org/10.5194/isprs-annals-IV-4-W6-49-2018

Portele, C. and Vretanos, P., 2018. OGC Web Feature Service 3.0-Part 1: Core. Retrieved from http://docs.opengeospatial.org/DRAFTS/17-069r1.html (June 2019)

Portele, C., 2019. OGC Testbed-14 Next Generation APIs: Complex Feature Handling Engineering Report. Retrieved from http://docs.opengeospatial.org/per/18-021.html (June 2019)

Webber, J., Parastatidis, S., Robinson, I., 2010. REST in practice: Hypermedia and systems architecture. O'Reilly Media, Inc.".

W3C., 2010. Same Origin Policy. Last accessed on April 2019.

Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubauer, A., 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), 5.

Zhu, W., Simons, A., Wursthorn, S., Nichersu, A., 2016. Integration of CityGML and Air Quality Spatio-Temporal Data Series via OGC SOS. In Proceedings of the Geospatial Sensor Webs Conference (GSW), Munster, Germany (pp. 29-31