MIGRATION OF DIGITAL CARTOGRAPHY TO CITYGML; A WEB-BASED TOOL FOR SUPPORTING SIMPLE ETL PROCEDURES

F. Fissore [1], F. Pirotti [1]

[1] Interdepartmental Research Center of Geomatics (CIRGEO), TESAF Department, University of Padova – (francesco.pirotti,francesca.fissore)@unipd.it

Commission IV, WG IV/4

KEY WORDS: CityGML, GIS, Extract Transform and Load, ETL, Open Source

ABSTRACT:

Digital cartography is notably produced in all countries, in different scales and formats. Latest cartographic production aims at creating 3D objects with topological consistency and rich information linked by attribute tables, i.e. the principles behind data to be managed in geographic information systems (GIS) environments. These data contain all the information necessary for production of the first levels of detail (LOD) of the CityGML model. The work presented reports on the first steps for a guided workflow to upload cartographic data containing building footprints, heights and other information, and migrating it to a validated CityGML model. The steps include a web-portal for uploading the data in a compressed archive containing shapefiles, and a back-end Python script that reads coordinate vertices, attributes and other necessary information, and creates a CityGML file. The process was tested on the Italian topographic geodatabase of some of the main cities of Italy. Discussion on workflow steps and results are presented. Results show that this process is feasible and it can be used to facilitate first tests on transforming existing cartography to CityGML models, which can be then used for further analysis.

1. INTRODUCTION

Three-dimensional capabilities in geographic analysis is nowadays a common concept. The paradigm shift from working with 2D or semi-3D representation to total 3D representation is now possible due to faster graphic cards and computing speed. Together with significant worldwide investments in network speed, these factors are contributing in bringing a lot of the processing effort to remote servers. This led to this investigation, i.e. the initial implementation of a web-based tool for creating 3D models from existing cartographic data. There are many data which are not born as 3D data, but do contain 3D information (e.g digital cartography). The biggest obstacle to fully exploit this information is variety and complexity of the source data formats. There is not a “one rule for all”, since digital cartography comes in a variety of formats and schemes.

Within the Urban-Geo Big Data project (Brovelli M.A., Boccardo P., Crespi M., Lanari R., Pirotti F., 2017), an Italian project of national interest (PRIN 2015), a large amount of cartographic data related to some of the main Italian cities was collected. The challenge to convert them to a CityGML model is exquisitely an Extract, Transform and Load (ETL) problem.
The Urban-Geo Big Data project aims to develop innovative geographic information systems (GIS) methodologies and tools to exploit the integration of traditional geomatic data, Earth Observations (EO), GNSS data and statistical data with new user-generated contents for promoting a more effective management of urban resources and infrastructures.

In order to transform the large amount of data in useful information for future developments and applications, it is important to structure 3D data into a rigorous geometric and semantic data model. Moreover, to guarantee the homogeneity of the 3D model products, a standardized data model is necessary.
The use of open standards together with the use of Free and Open Source Software (FOSS) ensures the interoperability, replication and reutilization of the applications and the homogeneity of data.
Currently CityGML is an international standard of OGC (Open Geosaptial Consortium) for the representation and exchange of 3D city models (Gröger and Plümer, 2012). European cities, such as Berlin, Lyon, Rotterdam and Wien, have implemented CityGML models of the whole city and released these information as open data (Carrión et al., 2010a; “Cities around the world with open datasets,”; Kolbe, 2009).

CityGML is an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is an application schema for the Geography Markup Language version 3.1.1 (GML3), the extendible international standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211 (“CityGML | OGC”). CityGML has a geometric model and a thematic model. The thematic model of CityGML covers various classes like buildings, relief, transportation, land use, tunnels, bridges, vegetation, water bodies and city furniture (Gröger and Plümer, 2012). If an object or attribute is not included in the standard, but is required for a specific implementation, Application Domain Extensions (ADEs) can be used. Taking a definition directly from (“Application Domain Extensions (ADE),” n.d.): ”an ADE is defined in an extra XSD (XML Schema Definition) file with its own namespace. This file has to explicitly import the XML schema definition of the extended CityGML modules. ADEs can be defined by information communities, which are interested in specific application fields. ADEs are increasingly being used in creating application specific extensions like for energy modelling, modelling topographic data, indoor modelling, noise modelling, etc.”. This provides a very high degree of customization and scalability of the model with several advantages which are well reported in (Agugiaro, 2016; Agugiaro et al., 2018).

In 3D geo-information science there are many different formats for saving 3D city models. This variety is a limit to the interoperability and the exchange of 3-dimensional data. This matter is a topic actually discussed in the 3D geoinformation community (Julin et al., 2018; Kolbe, 2009; Stadler et al., 2009) as standardization is a delicate topic. A new standard might add specific functionalities and work better than existing ones, but adds entropy unless it is easy for end users to apply ETL between the new standard and existing ones. This is the basis for the success of entrepreneurs that develop robust tools for ETL; e.g. software such as FME, which "makes integration of complex data a simple task".

In literature, there are several reports that describe methods for reconstructing 3D CityGML model from existent 3D models of geographic data, and even from 2D model. An example of predicting building height when height is not available is presented in (Biljecki et al., 2017): authors use ten predictors and a machine learning algorithm (random forest) to estimate heights of buildings. Results have a mean absolute error of 0.8 m and standard deviation of 1.8 m, showing possibility to generate a 3D city model from 2D data without elevation measurements. .
(Biljecki and Arroyo Ohori, 2015) investigate the automatic conversion between OBJ and CityGML. Both formats are commonly used to describe a 3D city model, although they are two different formats with different information. A method of automatic translation between the two formats is therefore interesting to investigate in order to enrich each model with information inherited from the other; in our study case the two formats are ESRI Shapefiles and CityGML.

CityGML is a multiscale model with 5 well-defined consecutive Levels of Detail (LOD) that contain increasingly detailed information. Each LOD has "sub-levels" that provide more information on the details that are necessary to go from one LOD to the next. Theoretically, the LOD with higher detail is very similar with the IFC format, which is an open file format used for Building Information Modelling (BIM). Each model comes from different worlds, therefore the integration is not automatic and seamless yet, but consistent work is ongoing towards this aim ("BIM to GIS (Advanced) | IFC LOD 200 to LOD 3 CityGML - FME Knowledge Center," n.d.; de Laat and van Berlo, 2011; Jusuf et al., 2017). IFC provides a model of a building or facility, including spatial elements, materials, and shapes.

(Donkers et al., 2016) investigated the possibility to improve the existing conversion algorithms from IFC to CityGML. They present a new algorithm that accurately applies the correct semantics from IFC models and that constructs valid CityGML LOD3 buildings by performing a series of geometric operations in 3D.

With the increased capability of sensing the surrounding world through portable sensors, e.g. smartphones, a great deal of geographic data generated by users became available, so-called "crowd-sourced" or "collaborative" data (Brovelli et al., 2018). Also OpenStreetMap (OSM) is considered a prominent example of volunteered geographic information aimed at the creation of a free editable map of the world. (Over et al., 2010) investigates the possibility to generate 3D city model from the OpenStreetMap (OSM) project and public domain height information provided by the Shuttle Radar Topography Mission. Moreover, they have implemented a web server to display 3D data. Finally (Sengul, 2012) addresses the problem of creating a CityGML model from aerial stereo images. Traditionally, acquisition of 3D objects is done by photogrammetric methods. The author therefore proposes a FME Workbench able to cover the gaps between the photogrammetric methods and CityGML. In this paper we investigate the possibility convert data from ESRI shapefile format to 3D CityGML model. Specifically, we are interested to automatically convert Italian cartographic data in 3D city model.
Quality cartographic products have always been in three dimensions. In the past they were delivered as graphical CAD-type formats; no topologic consistency was usually present or required. Recently, GIS-type formats are asked from producers, due to the augmented use of GIS software. Recent guidelines strictly require topologic consistency. Since about ten years ago, the creation of cartographic products in Italy, at regional scale (1:10000 to 1:1000), follows specific guidelines which are derived from the INSPIRE directive 2007/2/CE of 14 March 2007. The INSPIRE directive aims at a common spatial framework in Europe. Among the numerous points of the directive, an important indication is that all features must be topologically coherent in three dimensions. In other words, no gaps can be present and all nodes must correspond to other nodes of neighbouring features, unless at the periphery of the dataset. The cartography produced is often referred to as topographic geo-database, because of its definite orientation towards a database-like structure, with features and linked attributes. Other information is provided as attributes of features. One important information for deriving CityGML is the heights of features that have a significant thickness above the terrain surface plane, such as buildings. Height of building is usually referred to as the distance between the foot of the building and the bottom of the roof (practically the eaves). Other ancillary information can be added to features, enriching the contents of the topographic geo-database in production phase or later. For the above reasons, a topographic geo-database provides a rich set of information, which can be used to create CityGML models at LOD 1 or LOD 2 depending on the scale of the product and how accurate the data are.

Having a 3D model of the urban environment is important not only for viewing the 3D reality of the real world, but for analysts to apply methods that take into account the volumes of buildings and not just the footprints. For instance modelling solar irradiance requires detailed and accurate knowledge of the surfaces that intersect the sun rays (Piragnolo et al., 2015). The impact of new buildings can be assessed also in terms of impact of anthropic footprint on the environment (Piragnolo et al. 2014), which is gaining importance for planning mitigation strategies and support decisions.

In the following section we present a software able to convert cartographic data, containing building information, from ESRI shapefile format to 3D CityGML model. The software consists in two parts: a web tool, "*ZcityGML*" and a command line version, "*shp2city*", developed for practitioners. The paper is structured as follows: in section 2 the infrastructure used to develop the software and the architecture of front end and back end is presented; in section 3 the preliminary results of the software applied to Italian digital cartography and validation of CityGML model obtained in output using the tool webs "Shema" and "val3Dity" is shown. Section 4 discusses results and suggests future research direction.

2. MATERIALS AND METHODS

2.1 Infrastructure

Wanting to test migration of cartographic data to CityGML, we developed "ZcityGML", a web tool, accessible online.

ZCityGML runs on main browsers such as: Firefox, Safari, Chrome. It is written in R and Python languages and uses open source libraries.
The tool has been developed following a modular structure in order to facilitate a multi-user software development, and flexibility in design such as augmentation or exclusion of plugins. Furthermore, the choice to not use an exclusive programming language, made it possible to use libraries that better adjust to the best solution of specific tasks.

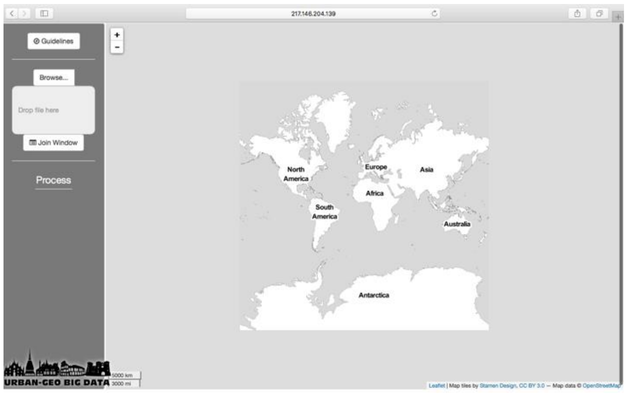
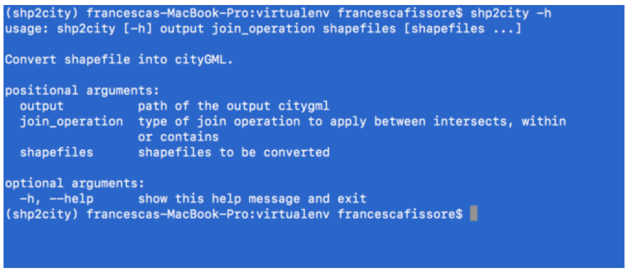


Figure 1. ZcityCML front-end.

2.1.1 Back End: Back end is written in Python and used mainly two of Pythonic open source libraries: (1) Geopandas (“GeoPandas 0.3.0,”.) and (2) lxml (“lxml”). GeoPandas is an open source project to make working with geospatial data in Python easier. It combines the capabilities of “Pandas” and “Shapely”, providing geospatial operations in pandas and a high-level interface to multiple geometries to shapely. GeoPandas was chosen because it enables to easily do operations on spatial data in Python that would otherwise require a spatially enabled database management system (DBMS) such as Postgresql/PostGIS. The lxml XML toolkit is a Pythonic binding for the C libraries libxml2 and libxslt. It is unique in that it combines the speed and XML feature completeness of these libraries with the simplicity of a native Python API, mostly compatible but superior to the well-known ElementTree API. This library was therefore used in the editing phase of the CityGML file.

The structure of back end of the tool web is a command line program without graphical interface that can work independently from the web interface. It is organized according to a modular structure that performs two fundamentals operation for the generation of the CityGML model. The first, starting from a collection of shapefile uploaded from the user, creates an “optimum” shapefile by spatial join operation. This step has been included in the case the properties of a geometry are distributed over several shapefiles. The second action is dedicated to the construction of the CityGML model starting from the geometries stored in the previously dataset and to write the 3D model in a CityGML file.

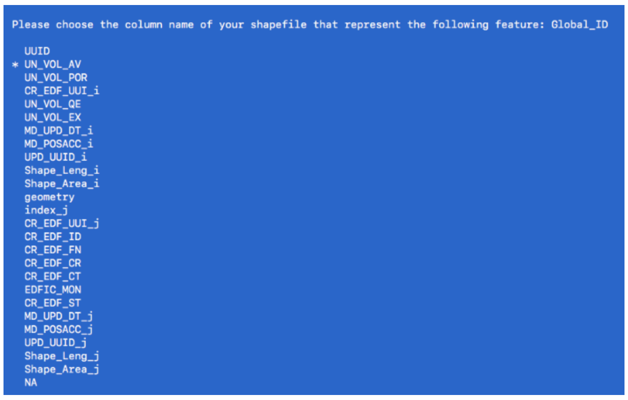
To perform these steps, several parameters should be set by user. Specifically, the following are requested: (1) the name of output CityGML file (2) a spatial relationship parameter, chosen between: “intersects”, “within” or “contains” and (3) paths of inputs files (see Figure 2). The first parameter will be used in the final step in order to save the 3D model. The second parameter is a fundamental request in order to perform spatial join between multiple files. The choice of this parameter is restricted to three possibilities: i) intersects: that returns true if the boundary and interior of each object intersects in any way with those of the other; ii) contains: true if each object’s interior contains the boundary and interior of the other object and their boundaries do not touch at all; iii) within: true if each object’s boundary and interior intersect only with the interior of the other (not its boundary or exterior), (inverse of contains). As for the remaining parameter, it can be composed of paths to one or two shapefiles. In case there is only one input file, the spatial join step is skipped and the CityGML file will be processed from the single input file purge from null geometries.



```
(shp2city) francescas-MacBook-Pro:virtualenv francescafissore$ shp2city -h
usage: shp2city [-h] output join_operation shapefiles [shapefiles ...]

Convert shapefile into cityGML.

positional arguments:
  output          path of the output citygml
  join_operation  type of join operation to apply between intersects, within
                  or contains
  shapefiles      shapefiles to be converted

optional arguments:
  -h, --help      show this help message and exit
(shp2city) francescas-MacBook-Pro:virtualenv francescafissore$ 
```

Figure 2. The help menu of shp2city.

In presence of two files, the spatial join process can be performed according to the spatial relation specified as input parameter. The first uploaded file is interpreted as target file, while the other ones as joined elements to the target. Moreover, before executing a spatial join operation input uploaded files are cleaned up by removing null geometries.



```
Please choose the column name of your shapefile that represent the following feature: Global_ID

  UUID
* UN_VOL_AV
  UN_VOL_POR
  CR_EDF_UUI_i
  UN_VOL_QE
  UN_VOL_EX
  MD_UPD_DT_i
  MD_POSACC_i
  UPD_UUID_i
  Shape_Leng_i
  Shape_Area_i
  geometry
  index_j
  CR_EDF_UUI_j
  CR_EDF_ID
  CR_EDF_FN
  CR_EDF_CR
  CR_EDF_CT
  EDFIC_MON
  CR_EDF_ST
  MD_UPD_DT_j
  MD_POSACC_j
  UPD_UUID_j
  Shape_Leng_j
  Shape_Area_j
  NA
```

Figure 3. Multiple selection: the user is asked to indicate which feature of the input shapefile meets a specific request.

Although CityGML defines ways to describe most of the common 3D features and objects found in cities (such as buildings, roads, rivers, bridges, vegetation and city furniture) and the relationships between them, in this preliminary phase, the CityGML model generated by our software will describe only 3D urban buildings at levels of detail LOD1. In order to develop the 3D model of city buildings, the user is required to indicate in which fields of the input dataset are stored some features essential to the creation of the 3D model (height of the building) and to enrich it (see Figure 3). Specifically, the features requested are:

- *Global ID*: unique identifier of single building.
- *Element ID*: unique identifier of single elements included in building.
- *Geometry height from terrain*: equivalent to the min height eaves.
- *Temporal validity*: the censum data of buildings,
- *Scale*: scale of representation.

- *Building category*: type of use for the building.
- *Building status*: condition of building (e.g erect, partially demolished etc…).

The requests for the global ID and the element ID arise from occurrence that one specific building can be composed of multiple building parts (e.g columns, pillars), described by geometries within the input shapefiles.
Using two different IDs, we are able to discriminate the entire building from individual architectonic structures. Each building will be uniquely identified through the Global ID (see Figure 4) and the structures that compose it with the Element ID (see Figure 5), like expected by the CityGML model.

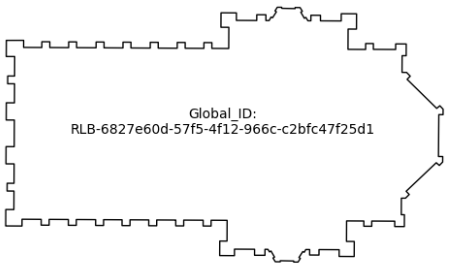


Figure 4. Example of Global ID of Milan’s Cathedral

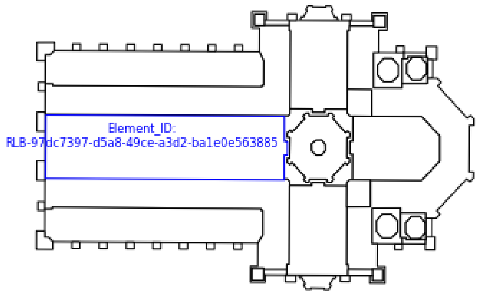


Figure 5. In blue an example of Element ID of a component of Milan’s Cathedral.

For every building or building part, the generalized outer shell is represented by exactly one prismatic extrusion solid. Ground, floor and roof surfaces must be horizontal, lateral boundary surfaces must be vertical. (“Modeling Guide for 3D Objects - Part 2: Modeling of Buildings (LoD1, LoD2, LoD3)”). In order to generate a prism, it is necessary to know its height. The height of each building in the input cartography is determined by the user, who selects the field of the attribute table of the shapefile that has the corresponding information related to the attribute "height from the ground". For example, for the LOD1 the height of prism equivalent to the eaves height. If the shapefile doesn't have the eaves height, a CityGML model will be generated at the LOD0 resolution. In other cases, the LOD1 level can be created. Regarded the horizontal plane that represent the ground of building, we set the minimum reference height as the minimum vertical coordinate of the geometry, since each vertex has its own height coordinate and they might not be the same for the building footprint. This is due to the production method of the cartography, which is based on photogrammetric methods, which provide a height value to each vertex when the producer is aligning the stereo-images.

About the remaining features, these are not essential to generate the geometry of the 3D model, but are additional attributes of interest which are important building information and must be migrated from the shapefile attribute table to the CityGML model.

2.1.2 Front End: There are two paradigms for processing data. One is to bring the software to the data, which is the common process of downloading the software in the client’s computer and proceeding with processing the data. The other way is to bring the data to the software by uploading the data to a remote computer with installed services,e.g. so called “SaaS”, Software as a Service. There is no best solution, as it depends on data size, hardware requirements, interaction that is necessary with the user and other factors, which are not the same for all cases. Due to investments in internet speed (e.g. optic fiber), and investments required to support high-end hardware, lately many users are considering the latter solution, e.g. send data to the software. The ETL process presented in this paper, considers both solutions. As mentioned in the introduction, the software consists in two parts: a web tool, "*ZcityGML*" and a command line version, “*shp2city*”, developed in Python. The actual ETL process is done with the back-end *shp2city* Python module. To make the process more accessible, the *ZcityGML* portal allows users to upload a zipped archive with shapefiles containing building features and information.

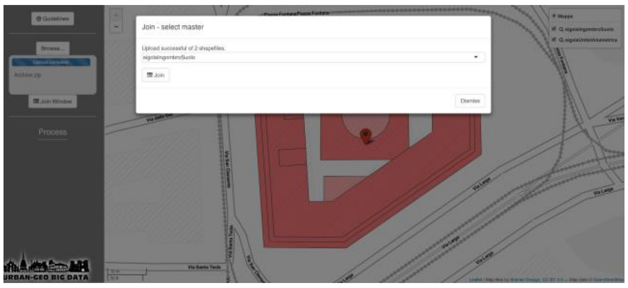
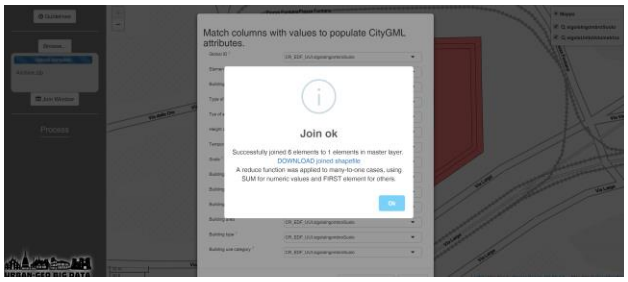


Figure 6. ZCityGML front-end after initial upload of a simple building complex.

Uploading shapefiles is done by simply dropping the zipped archive over the portal. The data is sent to the central server and extracted and inserted in a folder. Future implementations will provide a user with its personal disk space and folder structure to upload and organize its data. Depending on data size, the portal plots the data, or part of the data over a OpenStreetMap baselayer. Visualization supports the user into checking that the coordinate reference system is correctly read and guides the user into choosing the target layer for join operations, in case more than one layer is uploaded, as shown in Figure 6. Once the target layer is chosen, a spatial join assigns features to the target features and provides a set of columns from attribute tables of both the target layer and the joined layers. The result of the join operation can also be downloaded for further analysis (Figure 7).

The successive step is for the user to choose which attributes existing in the shapefiles’ tables can be included in the CityGML model (Figure 8). All the information inserted interactively by the user in the ZCityGML panel is then provided to the Python *shp2city* back-end to do the actual ETL process and provide the final CityGML file, which is compressed and provided to the user as a link for downloading. At this stage processing ends with CityGML produced server-side and downloading is not yet provided, as testing is still ongoing.

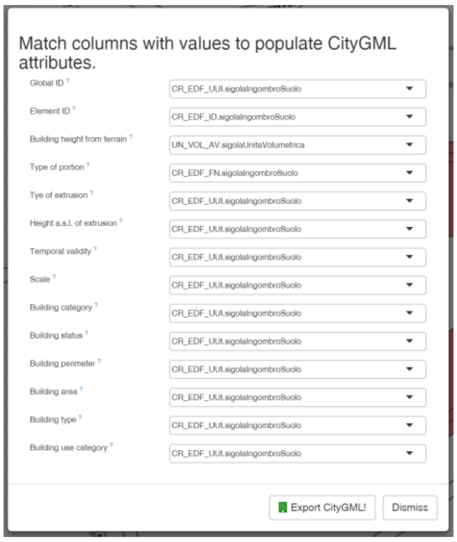


Match columns with values to populate CityGML attributes.

Join ok

Successfully joined 6 elements to 1 elements in master layer.
DOWNLOAD joined shapefile
A reduce function was applied to many-to-one cases, using SUM for numeric values and FIRST element for others.

Figure 7. ZCityGML after choosing target and joined layers, user is guided through next steps.



Match columns with values to populate CityGML attributes.

Global ID	CR_EDF_UUI.sigolaIngombroSuolo
Element ID	CR_EDF_ID.sigolaIngombroSuolo
Building height from terrain	UN_VOL_AV.sigolaUnitaVolumetrica
Type of portion	CR_EDF_FN.sigolaIngombroSuolo
Tye of extrusion	CR_EDF_UUI.sigolaIngombroSuolo
Height a.s.l. of extrusion	CR_EDF_UUI.sigolaIngombroSuolo
Temporal validity	CR_EDF_UUI.sigolaIngombroSuolo
Scale	CR_EDF_UUI.sigolaIngombroSuolo
Building category	CR_EDF_UUI.sigolaIngombroSuolo
Building status	CR_EDF_UUI.sigolaIngombroSuolo
Building perimeter	CR_EDF_UUI.sigolaIngombroSuolo
Building area	CR_EDF_UUI.sigolaIngombroSuolo
Building type	CR_EDF_UUI.sigolaIngombroSuolo
Building use category	CR_EDF_UUI.sigolaIngombroSuolo

Figure 8. Attributes from shapefiles can be linked to attributes of CityGML model to be created.

3. DISCUSSION

This section will show results related to testing ZCityGML web tool (version 1.0.1) and the back-end shp2city Python script, using cartographic data. Specifically, the results reported in following sections are referred to tests on loading specific layers which represent building elements in the topographic geo-database. The specific dataset is from the central zone of Milan and is available as open data from the online portal of the Lombardia Regione (“Home - Geoportale della Lombardia”). The portal provides download services of available databases in raster format or in vector format. Vector data consist of shapefiles with points, lines and polygons. Each element is associated with an alphanumerical database record containing all the attributes of the represented object. The table schemas in the database are structured according to regional guidelines. The topographic geo-database was downloaded and compressed files containing shapefiles for the different layers were extracted. Each file is expressed in the projected coordinate system UTM32N, referred to the geodetic reference system WGS84.
The two parts of the software (web portal and command line) were tested using as inputs two layers that belong to buildings: ‘Unità volumetrica’ and ‘Edificio’ which are respectively the footprints of the single building, defined as “volumetric unit” and the footprints of the building complex, defined as “edificio”; see Figure 9 for a depiction of the two layers.

The shapefiles from the two layers where archived in a compressed file and uploaded via the ZCityGML portal. For testing purposes only a few buildings were included. The spatial relationship ‘within’ has been chosen for the spatial join operation. The class ‘Unità volumetria’ was used as target file the spatial join operation. It is composed of 880 geometries, which together model the footprints the building. Attribute tables have twelve fields in total.

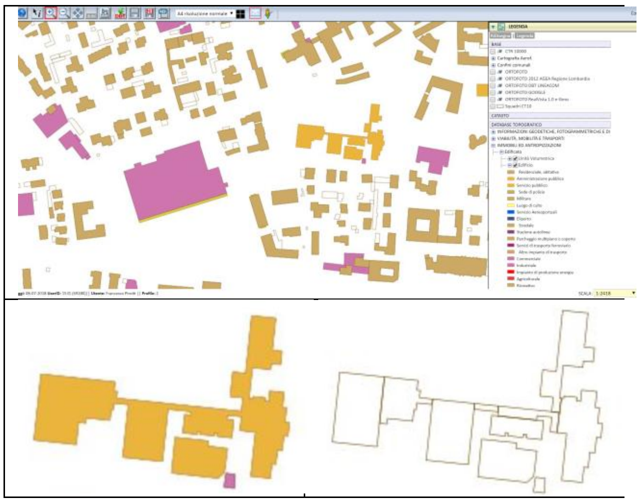


Figure 9. Top: representation of the two chosen layers with building representations. Bottom left: “edificio” (building). Bottom right: “unità volumetrica” (volumetric element).

Only some are of interest for migration to the CityGML model: the height of eaves (Geometry height from terrain), the scale of representation and the identification number of geometry (Element ID). The other class, “edificio”, was spatially joined to the target shapefile. This layer is made up of a smaller number of geometries with respect to the previous one (120 geometries), which represent the overall shape of the buildings complex. Among the attributes available in this layer, the used ones for migrating to the CityGML model are the following: the identification code of the geometries, used as global ID, the building census date, the type and status of the building complex. By monitoring the processing time needed to process the files, the spatial join operation was the one that required longer processing times, while the steps of processing the CityGML model and writing it to an output file were faster.

3.1 Validation

To validate the created CityGML model and fix errors of the process done by Python script *shp2city*, two external web tools were used. The tools were developed by the 3D geoinformation research group at TU Delft. The first tool is an online schema validator (Ledoux, 2018) and the second is a command-line tool, *val3Dity* (Ledoux, 2013). The first one is a web-application that validates a CityGML file against its XSD schemas. It requests as input a CityGML file and, if there are errors, the first one is reported. The tool can be reached by the link: (“CityGML schema validation”) and, the open source version is available on GitHub (*CityGML-schema-validation*, 2017) where it is possible to download a script. *Val3Dity* is a software able to validate 3D primitives according to the international standard ISO19107. Val3Dity is both a command-line program that web. The web tool is available at link (“val3Dity: geometric validation of GML 3D primitives”) and we have used it to validate the CityGML created by *shp2city*. The command-line and the source code are available on the repository Git-hub (*val3Dity*, 2018). Val3Dity generates an error report, available in two format (html or JSON) that allow the user to identify the features and the list of 3D

primitives with irregularity, along with a description of the problem.
In the validation phase of our output CityGML file, we have used the tool “Schema” in order to define the best XML structure to migrate digital cartography data to cityGML. Because the tool generates in output the first error, we have developed our xml tree step by step, testing every element. Finally using *val3Dity*, we have validated the primitive geometries used to model the buildings in a 3D space.

The validation procedures were used for fixing our errors in the processing steps carried out by the *shp2city* tool, related to the xml schema and 3D primitives. After this, the CityGML file produced automatically by the input shapefiles complies with the international standard ISO 19107.

3.2 Final output

At this stage of the project only partial area of the full cities have been converted to CityGML. Once the process is thoroughly tested with different sources (Padova, Milan, Rome, Turin etc.…) then the full building set will be transformed to CityGML and it will be provided as open data, with metadata, for downloading. GML files will be provided, but also a database (Postgres) will be used to store the models and interact with other partners of the project, which are preparing viewers and are also providing other data sources from different sensors. For example one partner will provide points from GNSS positions of bus tracks, so called floating car data, and another partner will provide raster data with binary information on soil sealing over the study areas. In this context CityGML is part of a larger structure that the project aims to create for better management of the urban environment.

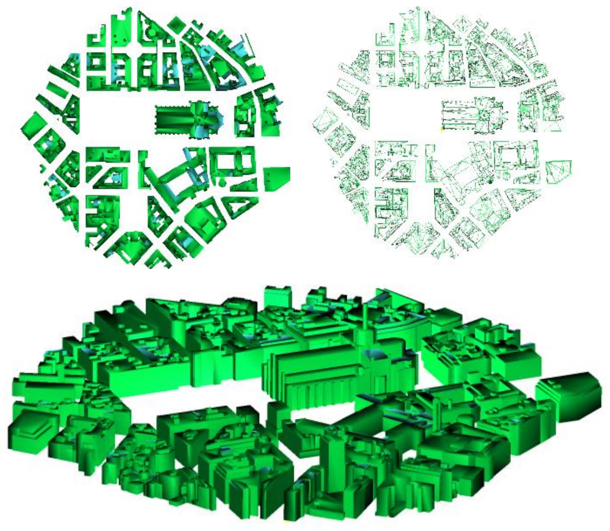


Figure 10. CityGML Model of Milan central square.

4. CONCLUSION AND FUTURE WORK

This article introduced the first version of the command line software “*shp2city*” able to convert vector geographical data to cityGML and the web front-end for uploading input data, ZCityGML. Shp2city will be further developed as part of Urban-Geo Big Data, an Italian project of national interest (PRIN 2015), which aims to develop innovative GIS methodologies and tools to exploit the integration of traditional geomatic data, Earth Observations (EO) and statistics data. To achieve the objectives of the project, it is necessary to develop a tool that, by implementing ETL procedures, is able to migrate digital Italian Cartographic to the CityGML model. This software has been developed using free and open source tools and will be provided in agreement to open source licenses.

The positive results obtained in this initial phase of developing and in the validation steps, thanks to third-party tools, as previously explained, confirm the possibility to transform the Italian cartography in topographic geo-database to 3D city models at LOD 1 or LOD 2 level. The biggest problems that can (and probably will) arise are topological errors that might still be present in the source data. Theoretically the source data should be topologically coherent, as each cartographic product should undertake a quality assessment procedure, but if that has not been done, the error will be propagated to the CityGML model. In this case the errors must be corrected by automatic or semi-automatic procedures. It is worth noting that ETL procedures can bring to light problems in the geometry or attribute schemas, and can thus help improve the existing data.

For future work we plan to investigate the possibility to integrate, in the process implemented in *shp2city*, the methods explained in (Biljecki et al., 2017) for the cartographic data without elevation. In addition we are interested to integrate the two validator software into our tool. This improvement would allow to generate the CityGML file and validate it simultaneously, providing the user precious information on errors that are present in its source data. Moreover, using the *val3Dity* tool reports we want to extractt the part of geometry affected by problems. This way, the user can be guided to an understanding of the nature of the error and to what part of the data is responsible for the error, making the resolution simpler.

Finally, from the analysis of the report produced by *val3Dity*, the occurrence of errors in case of badly formatted cartographic data emerged; issues not investigated in the first version of *shp2city*, but that we intend to solve in future versions.

ACKNOWLEDGEMENTS

This research is supported by the Italian Ministry of Education through the project of national interest (PRIN) Urban-Geo Big Data (project code: 20159CNLW8 - PE10).

REFERENCES

Agugiaro, G., 2016. Energy planning tools and CityGML-based 3D virtual city models: experiences from Trento (Italy). *Applied Geomatics*, 8, 41–56. https://doi.org/10.1007/s12518-015-0163-2

Agugiaro, G., Benner, J., Cipriano, P., Nouvel, R., 2018. The Energy Application Domain Extension for CityGML: enhancing interoperability for urban energy simulations. *Open Geospatial Data, Software and Standards*, 3. https://doi.org/10.1186/s40965-018-0042-y

Application Domain Extensions (ADE) [WWW Document], 2018 URL https://www.citygml.org/ade/ (accessed 7.2.18).

Biljecki, F., Arroyo Ohori, K., 2015. Automatic Semantic-preserving Conversion Between OBJ and CityGML. Eurographics. *Workshop on Urban Data Modelling and Visualisation* https://doi.org/10.2312/udmv.20151345

Biljecki, F., Ledoux, H., Stoter, J., 2017. Generating 3D city models without elevation data. *Computers, Environment and Urban Systems*, 64, 1–18. https://doi.org/10.1016/j.compenvurbsys.2017.01.001

BIM to GIS (Advanced) | IFC LOD 200 to LOD 3 CityGML - FME Knowledge Center [WWW Document], 2017. URL https://knowledge.safe.com/articles/1024/bim-to-gis-advanced-ifc-lod-200-to-lod-3-citygml.html (accessed 7.5.18).

Brovelli M.A., Boccardo P., Crespi M., Lanari R., Pirotti F., 2017. URBAN GEO-BIG DATA - URBAN GEOmatics for Bulk Information Generation, DAta assessment and Technology Awarness [WWW Document]. URBAN GEO-BIG DATA. URL http://www.urbangeobigdata.it/ (accessed 7.7.18).

Brovelli, M.A., Minghini, M., Zamboni, G., 2018. New Generation Platforms for Exploration of Crowdsourced Geo-Data, in: Mathieu, P.-P., Aubrecht, C. (Eds.), *Earth Observation Open Science and Innovation*. Springer International Publishing, Cham, pp. 219–243. https://doi.org/10.1007/978-3-319-65633-5_9

Carrión, D., Lorenz, A., Kolbe, T.H., 2010. Estimation of the Energetic Rehabilitation State of Buildings for the City of Berlin Using a 3D City model Represented in CityGML. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XXXVIII-4/W15

Cities around the world with open datasets [WWW Document], n.d. URL https://www.citygml.org/3Dcities/ (accessed 7.2.18).

CityGML | OGC [WWW Document], URL http://www.opengeospatial.org/standards/citygml (accessed 7.2.18).

CityGML schema validation [WWW Document], n.d. URL http://geovalidation.bk.tudelft.nl/schemacitygml/ (accessed 6.25.18).

CityGML-schema-validation: small script to validate a CityGML file against the XSD schemas of CityGML (all versions supported), 2017. 3D geoinformation research group at TU Delft.

de Laat, R., van Berlo, L., 2011. Integration of BIM and GIS: The Development of the CityGML GeoBIM Extension, in: Kolbe, T.H., König, G., Nagel, C. (Eds.), *Advances in 3D Geo-Information Sciences*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 211–225. https://doi.org/10.1007/978-3-642-12670-3_13

Donkers, S., Ledoux, H., Zhao, J., Stoter, J., 2016. Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings. *Transactions in GIS*, 20, 547–569. https://doi.org/10.1111/tgis.12162

GeoPandas 0.3.0 [WWW Document], n.d. URL http://geopandas.org/ (accessed 6.5.18).

Gröger, G., Plümer, L., 2012. CityGML – Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing* 71, 12–33. https://doi.org/10.1016/j.isprsjprs.2012.04.004

Home - Geoportale della Lombardia [WWW Document], URL http://www.geoportale.regione.lombardia.it/en/home;jsessionid=055A0E6B4225DE36604E759FB70CA7D2 (accessed 5.31.18).

Julin, A., Jaalama, K., Virtanen, J.-P., Pouke, M., Ylipulli, J., Vaaja, M., Hyyppä, J., Hyyppä, H., 2018. Characterizing 3D City Modeling Projects: Towards a Harmonized Interoperable System. *ISPRS International Journal of Geo-Information*, 7, 55. https://doi.org/10.3390/ijgi7020055

Jusuf, S.K., Mousseau, B., Godfroid, G., Soh, J.H.V., 2017. Path to an Integrated Modelling between IFC and CityGML for Neighborhood Scale Modelling. *Urban Science*, 1, 25. https://doi.org/10.3390/urbansci1030025

Kolbe, T.H., 2009. Representing and Exchanging 3D City Models with CityGML, in: *3D Geo-Information Sciences, Lecture Notes in Geoinformation and Cartography*. Springer, Berlin, Heidelberg, pp. 15–31. https://doi.org/10.1007/978-3-540-87395-2_2

Ledoux, H., 2018. CityGML schema validation [WWW Document]. URL http://geovalidation.bk.tudelft.nl/schemacitygml/ (accessed 6.25.18).

Ledoux, H., 2013. On the Validation of Solids Represented with the International Standards for Geographic Information. *Computer-Aided Civil and Infrastructure Engineering*, 28, 693–706. https://doi.org/10.1111/mice.12043

lxml [WWW Document], URL http://lxml.de/ (accessed 6.5.18).

Modeling Guide for 3D Objects - Part 2: Modeling of Buildings (LoD1, LoD2, LoD3) - SIG3D Quality Wiki EN [WWW Document], n.d. URL http://en.wiki.quality.sig3D.org/index.php/Modeling_Guide_for_3D_Objects_-_Part_2:_Modeling_of_Buildings_(LoD1,_LoD2,_LoD3)#Heights (accessed 6.12.18).

OpenStreetMap [WWW Document], n.d. . OpenStreetMap. URL https://www.openstreetmap.org/ (accessed 7.4.18).

Over, M., Schilling, A., Neubauer, S., Zipf, A., 2010. Generating web-based 3D City Models from OpenStreetMap: The current situation in Germany. *Computers, Environment and Urban Systems, GeoVisualization and the Digital City* 34, 496–507. https://doi.org/10.1016/j.compenvurbsys.2010.05.001

Piragnolo, M., Masiero, A., Fissore, F., Pirotti, F., 2015. Solar Irradiance Modelling with NASA WW GIS Environment. *ISPRS International Journal of Geo-Information*, 4, 711–724. https://doi:10.3390/ijgi4020711

Piragnolo, M., Pirotti, F., Guarnieri, A., Vettore, A., Salogni, G., 2014. Geo-Spatial Support for Assessment of Anthropic Impact on Biodiversity. *ISPRS International Journal of Geo-Information*, 3, 599–618. https://doi:10.3390/ijgi3020599

Sengul, A., 2012. Extracting Semantic Building Models from Aerial Stereo Images and Conversion to CityGML. *ISPRS - International Archives of the Photogrammetry, Remote Sensing* *and Spatial Information Sciences* XXXIX-B3, 321–324. https://doi.org/10.5194/isprsarchives-XXXIX-B3-321-2012

Stadler, A., Nagel, C., König, G., Kolbe, T.H., 2009. Making Interoperability Persistent: A 3D Geo Database Based on CityGML, in: Lee, J., Zlatanova, S. (Eds.), *3D Geo-Information Sciences*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 175–192. https://doi.org/10.1007/978-3-540-87395-2_11

val3Dity: geometric validation of GML 3D primitives [WWW Document], URL http://geovalidation.bk.tudelft.nl/val3Dity/ (accessed 6.26.18).