IFC CLASS ASSIGNMENT ADD-ON: BIM MODELLING FOR ARCHITECTURAL AND ENGINEERING PROJECTS USING BLENDER

Ana Pérez-García[1], Norena Martín-Dorta[1], José Ángel-Aranda[2]

[1]Universidad de La Laguna (Spain)

[2]Universitat Politècnica de València (Spain)

Received: 28/jul/2023– Reviewed: 02/aug/2023 – Accepted: 06/jan/2024 – DOI: https://dx.doi.org/10.52152/D11029

To cite this article: PÉREZ-GARCÍA, Ana; MARTÍN-DORTA, Norena; ARANDA, José-Ángel. IFC CLASS ASSIGNMENT ADD-ON: BIM MODELLING FOR ARCHITECTURAL AND ENGINEERING PROJECTS USING BLENDER. DYNA. September October 2024. vol.99, n.5, pp. 481-486. DOI: https://dx.doi.org/10.52152/D11029

ABSTRACT:

Building Information Modelling (BIM) is a working methodology for the design and management of construction projects that facilitates collaboration and communication between the different parties involved. It is based on the digital three-dimensional representation of infrastructures within the field of architecture and civil engineering. The generated models are true data containers, capable of storing the geometric and semantic information necessary to support the design, construction and maintenance operations of the assets. This methodology promotes data exchange between different software tools through interoperability and the use of standards. Usually in the development of projects throughout their life cycle, commercial, closed-source BIM software is used. This generally leads to limitations in interoperability and data reuse. This paper details advances in the use of Blender, a free and open-source software (FOSS), to support BIM workflows in construction projects. The paper addresses the automation of the process to export a 3D Blender model to a BIM model in IFC (Industry Foundation Classes). The IFC Class Assignment add-on is developed, which allows the automated assignment of IFC classes to the different elements of the 3D model in Blender to convert it into a BIM model, using scripts written in Python. The proposed extension is able to generate, in an agile and simple way, digital models in IFC4, adequately storing the geometric and semantic information of its components. The results provide a tool that contributes to automate the creation of BIM models in an open-source software such as Blender, simplifying modelling processes and supporting collaboration and standardisation in architectural and engineering projects.

Keywords: BIM, IFC, Automation, Free and Open-Source Software, Blender, Python.

1. - INTRODUCTION

BIM models are centralised data repositories that can be used to share reliable information about an asset throughout its life cycle. Their use allows a project to be designed directly in 3D and completely in its entirety, defining elements or objects in compliance with specific information standards. As a result, both the initial drafting of the project and potential changes, if necessary, can be carried out more efficiently and quickly [1].

According to ISO 37100 [2], interoperability is the ability of systems to provide services to and accept services from other systems and to use the services so exchanged to enable them to operate effectively together. Within the framework of BIM methodology, interoperability facilitates the exchange of information between different software tools, which can be consulted by the different parties involved in the life cycle of the assets, both for their visualisation and their transformation. Related to interoperability is the concept of openBIM. It is a collaborative process approach, based on open standards and workflows that aims to improve the quality of buildings and infrastructure projects and assets [3]. OpenBIM standards, promoted by the international organisation buildingSMART, facilitate a transparent and open workflow that enables the participation of all stakeholders in a project. It also creates a common language for widely used processes, avoiding multiple entries of the same data and the resulting errors. There are several openBIM standards, but the most widely used is Industry Foundation Classes (IFC). It is an open and neutral, non-proprietary data schema, developed by buildingSMART and regulated in ISO 16739-1, which describes the built environment, including buildings and civil works, in a digital and standardised way [4].

The employment of interoperable formats ensures workflow continuity, enabling users to link different tools and systems to facilitate multiple ways of collaboration and exchange [5]. Therefore, it is necessary to develop procedures that minimise the effort required for the creation of BIM models and their export to this type of standards. This research work approaches the creation of BIM models by automating the geometric classification processes of models in the IFC data schema. As a novelty, the capabilities offered by a free and open-source software (FOSS) such as Blender and scripts developed in Python are used.

2. - LITERATURE REVIEW

As projects in the Architecture, Engineering, Construction and Operations (AECO) sector grow in size, companies need to develop new tools to generate graphical documentation, manage changes and update information during the design phase. Traditional CAD design software is being replaced by BIM methodology, which provides great advantages in terms of data integration, as well as reduced design costs and change orders, improved coordination between the different professionals and teams involved, increased productivity, and reduced design errors. It also requires innovative development of effective decision support tools based on BIM databases and strengthened cooperative work with data exchange capabilities [6].

Along with the advantages of using BIM, the automation of the different workflows that comprise the methodology must also be considered. There are many studies that address this issue. For example, Sena [7] manipulated the data of a project created with Revit (Autodesk, closed-source software) using the visual programming tool Dynamo and the languages Python and C#. This made it possible to optimise certain elements, obtain complex shapes, and minimise repetitive tasks. In this context, Janssen et al. [5] showed the feasibility of a semi-automated workflow for the generation of BIM models from minimal geometric models, consisting of explicitly defined building elements, but with limited information. Tests for the validation of the proposed flow were performed with the proprietary software Rhino, Houdini and ArchiCAD. It aims to simplify the modelling of standard building elements, allowing parties involved in the design to focus on the modelling of more complex elements.

While some studies use automation to streamline and optimise modelling processes, other focus on generating BIM models from an open-source software environment. A notable one is the work of Puech de Oriol [8]. The author developed, using Python, a set of five utilities that allow users to easily model certain entities from the field of civil engineering in Blender. In particular, the proposed tools enable the creation of Digital Terrain Models (DTM), longitudinal profiles of a road, pipes and natural and artificial breakwaters for harbours. However, a limitation of this work is that the presented utilities do not include the assignment of IFC classes to the modelled elements. Another example is that provided by Clemen et al. [9] Their work consisted in creating a free and open-source tool to export DTMs in various file formats (LandXML, CityGML, CAD/DXF, GRID...) to the version of the IFC schema required by the user, in order to promote the openBIM workflow. Meanwhile, Malewczyk [10] explored the concept of openBIM by examining the integration capabilities of Blender and BlenderBIM add-on with the IFC standard. According to the author, the combined use of Blender and BlenderBIM allows to treat the IFC standard as a native way of recording information, unlike most BIM software. This solution would improve collaboration on projects.

Despite the above, there is limited evidence in the scientific literature on the automation of BIM model generation with an open-source software. Therefore, this work addresses the possibilities offered by a tool such as Blender for this purpose and its customisation through the development of an add-on.

3. - MATERIAL & METHODS

This research work approaches the creation of an extension or add-on for the automation of BIM modelling processes in the Blender software environment and the export of the results to the IFC data schema. A menu is developed in Blender, through the configuration of different scripts in Python language. An important limitation currently existing is that assigning IFC classes to objects modelled natively in Blender or using an add-on is not an automatic process. For this reason, the proposed tool aims to streamline and enhance workflows, avoiding the need to manually assign IFC classes to the different entities that make up a basic architectural model.

3.1. - BLENDER

Blender is a free and open-source 3D design software. It supports both organic and parametric modelling and has a customisable OpenGL graphical interface with Python scripts for the development of extensions to add new functionalities to the application. Python is an interpreted, cross-platform, high-level programming language.

Executing code snippets in Blender can be done in two ways: (1) by using the program's built-in text editor, or (2) by entering commands in the Python console.

Modelling in Blender is done through a series of objects, consisting of two clearly differentiated parts:

- The modelled object: contains information about the position, rotation and size of a particular element.
- Object data: contains everything else. In case the object is composed of meshes, it stores the geometry, the material list, or the vertex groups. Each object has a link to its associated object data, and the same object data can be shared by many objects.

In this work two existing extensions for Blender are used in conjunction with each other: Archipack and BlenderBIM. Archipack is a database of parametric BIM objects for architectural modelling. It also has a real-time editing interface for the management of these objects [11]. BlenderBIM is an extension developed under the IfcOpenShell open-source library, which works as a complete BIM authoring platform, providing functions related to reading, writing and analysing BIM models [12], [13].

3.2. - IFC SCHEMA

The IFC schema codifies in a logical way the identity and semantics of objects, the characteristics or attributes of abstract concepts, and the relationships between processes and people [4]. The difference between IFC and other schemas resides in the fact that it is based on geometrically represented objects with information and relationships specific to the building domain. This provides a structure that supports the use of information and the development of a strategy for the operation and management of the asset. However, it should also be noted that the elements of BIM models do not always have a defined geometry, in which case their representation in the IFC schema is purely semantic [14].

The versions of the schema currently in use are IFC2x3, IFC4 and IFC4x3. IFC files can contain different kinds of elements, classified into entities (classes or categories of objects), types and properties (Fig. 1). Entities are the main elements of the schema, and can be either physical or abstract. Types are the basic metadata of an entity, and include the direct and inverse relationships that exist with other elements. Finally, properties constitute complementary information to an entity, and are grouped in so-called Property Sets (Psets).

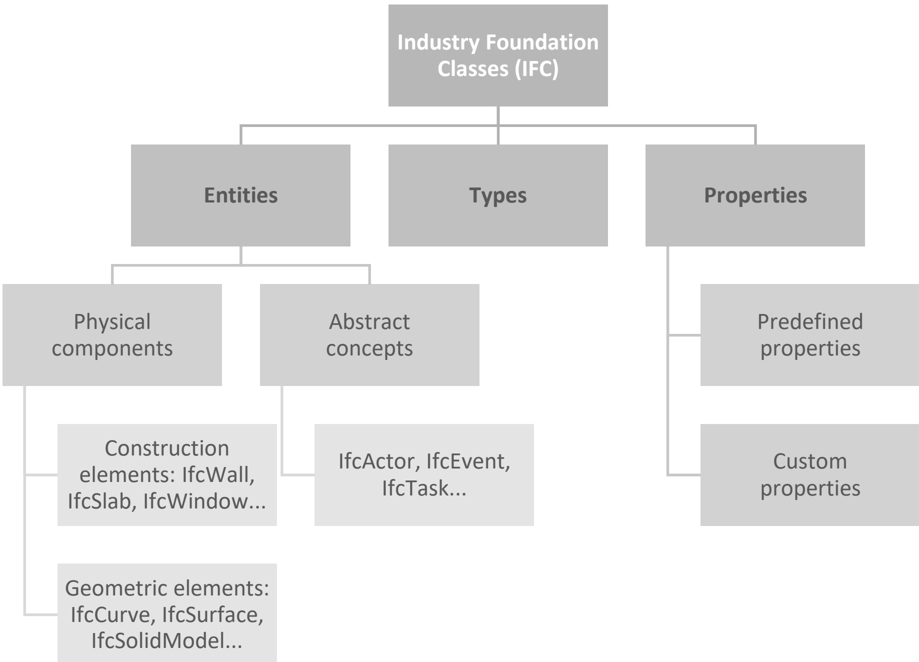

Fig. 1. Components of the IFC schema.

The schema version used in the development of this work is IFC4 ADD2 TC1. The basic objects that can be found in IFC are the spatial structure elements, such as site, building, building storey or space, and the building elements (walls, slabs, stairs, etc.) [14].

One limitation of Blender is that it lacks native functionality for mass-assigning IFC classes to modelled objects, but this can be achieved by using the IfcOpenShell Python library or alternatively the BlenderBIM add-on. However, as mentioned above, this assignment of classes to objects created directly with Blender or imported from extensions developed specifically within its environment, such as Archipack, is not automated. It has to be done manually, element by element, or by filtering object types, which involves a significant amount of time when dealing with models of a certain size.

3.3. - PROPOSED TOOL: IFC CLASS ASSIGNMENT ADD-ON

The add-on developed in this work aims to simplify the process of creating and exporting an architectural model created in Blender to the IFC data exchange scheme. Its menu allows an automated selection of the different elements (walls, slabs, columns, ...) that integrate the model, assigning them the corresponding building level within the IFC structure, as well as the appropriate class. It is designed to serve as a connection between the existing add-ons Archipack and BlenderBIM, making use of the functionalities of both, and is optimised for Blender version 3.5. Fig. 2**¡Error! No se encuentra el origen de la referencia.** shows the workflow developed for the creation of BIM models in the IFC schema.

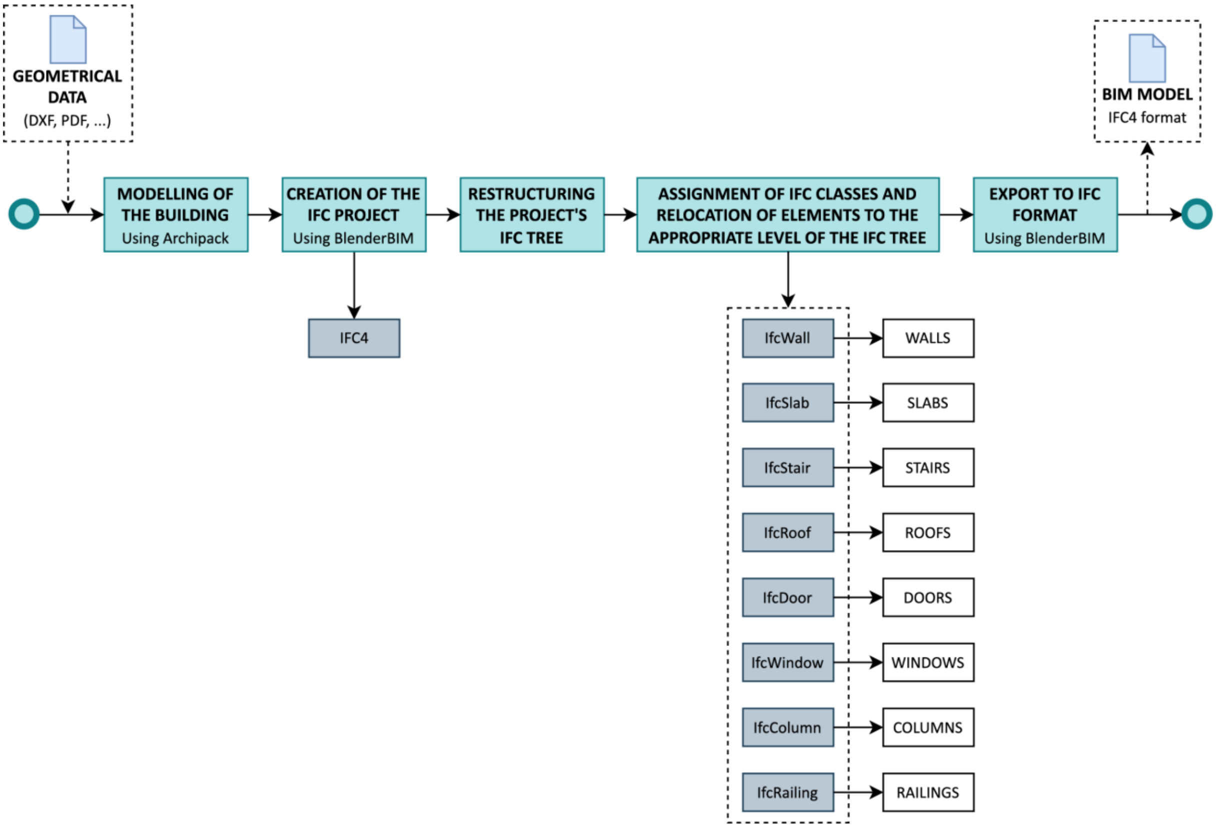

Fig. 2. Functional diagram of the IFC Class Assignment add-on.

Parametric modelling is done using the Archipack add-on. It is possible to start by importing the geometrical information of the building into Blender, using it as a basis for the modelling, in vector or raster format. A reference object is assigned to each floor, which will contain the information related to its location and elevation (Fig. 3**¡Error! No se encuentra el origen de la referencia.**). This will later be used to classify the elements by levels within the IFC schema. The different spaces are modelled using horizontal and vertical parametric objects: walls and slabs, respectively, and the different carpentry elements (doors and windows) are positioned. Finally, the pillars, the interior staircase and the railing of the roof parapet are placed. Once the model is complete, the IFC project is created using BlenderBIM functionalities through the proposed extension—it works correctly under the IFC4 version of the standard. This generates in the Blender Outliner menu a project tree with the basic structure of the building model in the IFC schema. The developed tool allows its restructuring, adding automatically to the tree as many levels as number of floors the building has (Fig. 3**¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.**). If during the modelling with Archipack the elements have been classified at the appropriate level, the proposed extension detects the floor to which they belong and relocates them to the corresponding level of the IFC tree when assigning classes. This is an advantage over the BlenderBIM workflow, where restructuring the IFC tree has to be done manually.

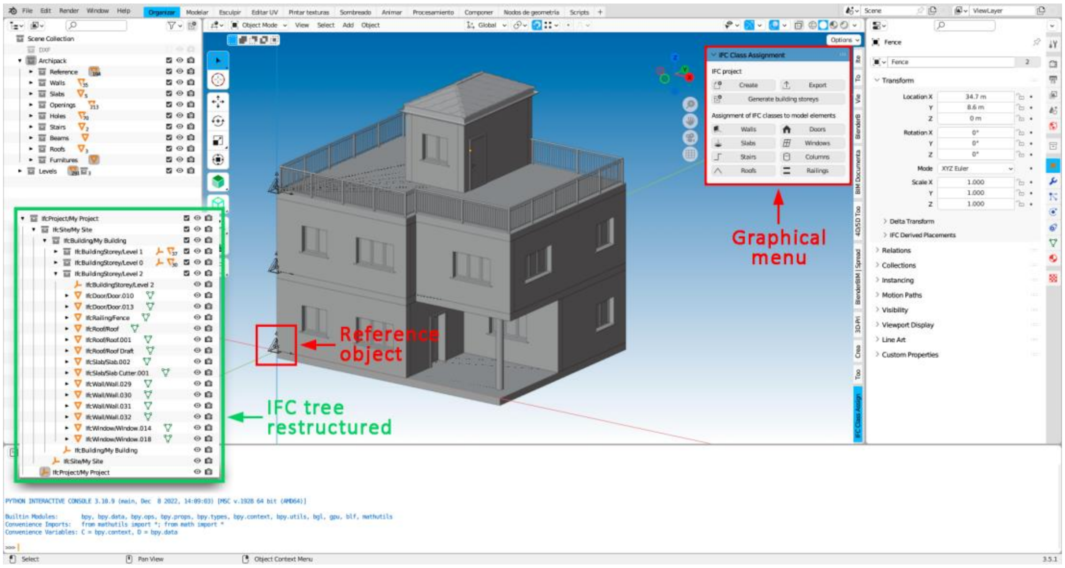

Fig. 3. Architectural model generated with Archipack.

The developed extension, IFC Class Assignment, consists of two Python scripts: *operators.py* and *panel.py*. The first one contains the Python classes needed to restructure the project tree and the assign the IFC class to each of the elements that integrate the model, by invoking the corresponding BlenderBIM functionalities. The second script constitutes the graphical menu of the add-on (Fig. 4**¡Error! No se encuentra el origen de la referencia.**). The extension contains a total of eleven functionalities presented in a menu with buttons (Fig. 3**¡Error! No se encuentra el origen de la referencia.**). These buttons are calls to the different Python classes created. The classes consist of a set of functions, developed based on the task intended to be performed by the software.

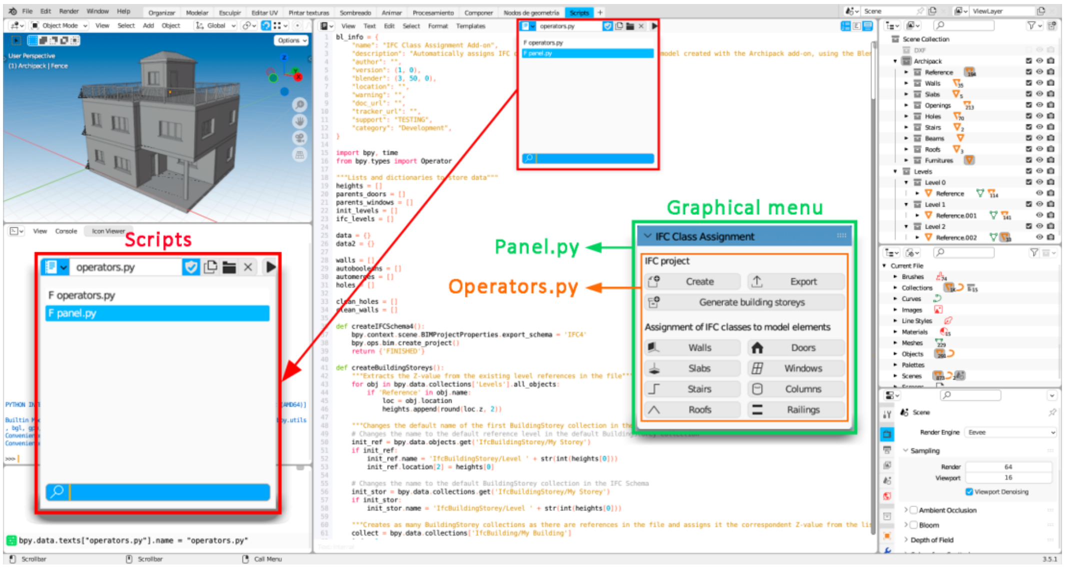

Fig. 4. Python scripts developed for the creation of the add-on.

Create and *Export* functions make it possible to create the IFC structure within the Blender file currently in use and to export it. *Generate building storeys* function can modify the level structure of the IFC tree, adapting it to the generated model. On the other hand, the buttons *Slabs*, *Stairs*, *Roofs*, *Doors*, *Windows*, *Columns* and *Railings* enable the automated assignment of IFC class to these elements. *Walls* button, on the other hand, not only allows assigning IFC class to walls and partitions, but also links, in an automated way, door and window openings to the walls in which they are hosted, executing the corresponding Boolean difference operation.

4. - RESULTS

Fig. 5 shows the visualisation in a free viewer of the BIM model resulting from the IFC class assignment process and its export from Blender. During the creation of the add-on, some notable incidents have been recorded. The most relevant ones refer to (1) the assignment of IFC classes to carpentry elements—doors and windows—, and (2) the linking of carpentry openings with host walls.

The case of doors and windows is particular, since, unlike the rest of the elements, they are composed of a set of associated sub-entities (frames, panels, handles, etc.)**¡Error! No se encuentra el origen de la referencia.**. This generates incidents that make it impossible to assign the corresponding IFC class. This problem has been solved with the development of a function that automatically joins all the sub-entities into a single one**¡Error! No se encuentra el origen de la referencia.**.

On the other hand, it is not enough to assign IFC class to walls, doors and windows for their correct visualisation in the schema. It is also necessary to link their respective holes or openings to the walls that contain them. Otherwise, as a result, doors and windows will be embedded within the walls**¡Error! No se encuentra el origen de la referencia.**.

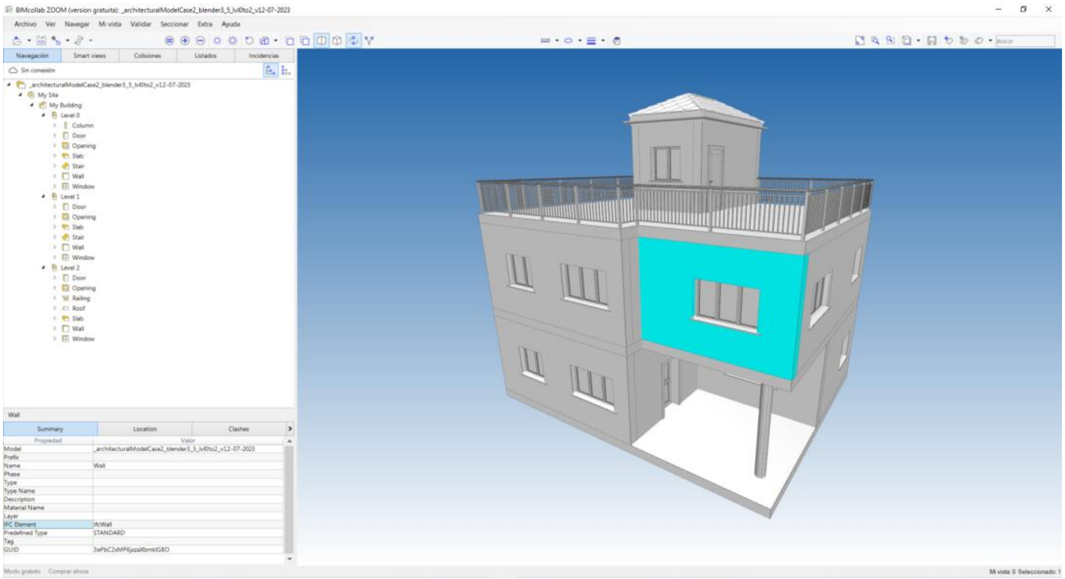

Fig. 5. Visualisation of the BIM model in a free IFC viewer.

The parametric objects that Archipack integrates are constituted by meshes. These are made up of three basic structures: vertices, edges and faces. Faces can be defined by three (triangles), four (quads) or more (n-gons) vertices. BlenderBIM makes use of the function createIfcRelVoidsElement(), contained in the Python library IfcOpenShell, to create the relationship between holes or voids and walls or partitions. During this process, it has been observed that the meshes that constitute the geometric representations of certain walls in the models used as a test in Blender have disappeared (Fig. 6). The error analysis has revealed that certain carpentry holes of similar typology have the same data source. This is because the same object data can be shared by many objects, as explained in section 3.1.

When creating the link between a hole and its wall, the geometric representation of the first object is integrated into the second, becoming a single element. As a result, the rest of the openings that shared the same attributes lose their data source. If this happens, the software is unable to apply the Boolean difference operation correctly when repeating the linking process between holes and walls. Consequently, the wall mesh is lost in its transformation to Constructive Solid Geometry (CSG). To solve this error, the developed code includes a function that eliminates the duplicity of data, creating a unique source of information for each of the carpentry holes contained in the model.

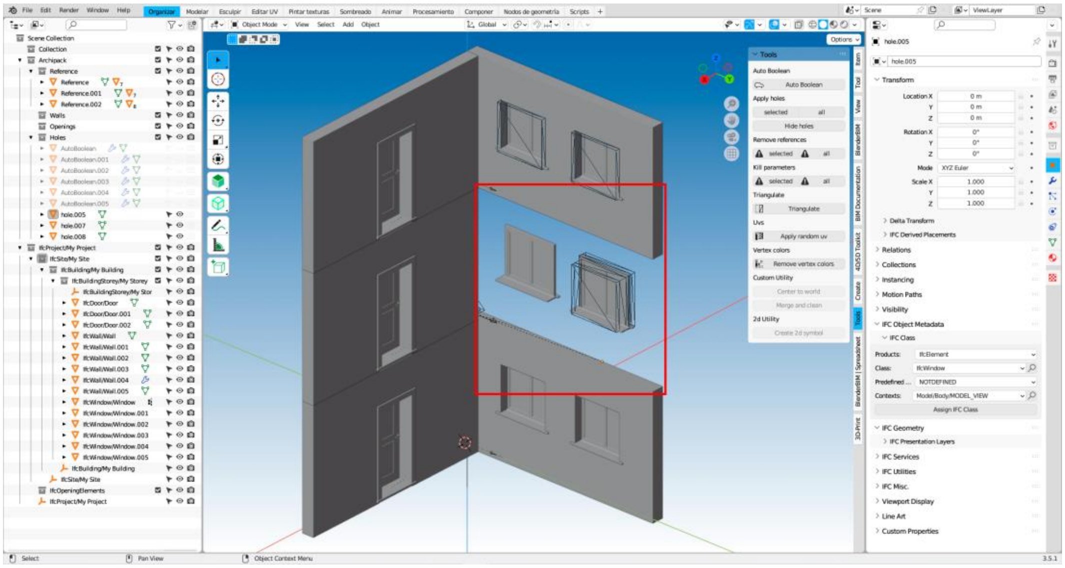

Fig. 6. Disappearance of the geometrical representation of one of the walls on the second level of the model.

5. - CONCLUSIONS

This paper demonstrates the usefulness of using a free and open-source software (FOSS) such as Blender for the automated modelling of building elements and their export as a BIM model to the IFC data schema for the impulse of collaborative work in an openBIM environment. The proposed add-on extends the basic functionality of Blender by displaying a fully customisable menu through which the software can be run. A code is developed that allows to generate in an automated process the IFC structure within the modelling platform, to restructure it by levels, to assign IFC classes—through BlenderBIM—to the different elements created with Archipack, and to link the carpentry holes to the different walls and partitions to which they belong with a Boolean difference operation, for the correct visualisation of openings in the BIM model.

In this case, the programmed code allows the integration in an open-source software of the definition of complex geometries from basic elements and their export to an open and neutral exchange format, enabling models to be visualised afterwards by the different parties involved in the process through an openBIM workflow. This is done using simple interfaces that support agile interaction with the models. Previously, the procedure of assigning IFC classes to the components of a model in the context of openBIM had to be done manually. The proposal presented in this work provides a simplification of the BIM modelling processes, promoting continuity in the flow of information. The IFC Class Assignment add-on has been developed as part of an academic research project. It has been tested with Blender 3.5 and is limited to the IFC4 ADD2 TC1 version of the standard.

REFERENCES

[1] Chang, K. M., Dzeng, R. J., Wu, Y. J. "An automated IoT visualization BIM platform for decision support in facilities management". *Appl. Sci.* 2018. Vol. 8. No. 7. DOI: 10.3390/app8071086

[2] International Organization for Standardization (ISO). "ISO 37100:2016 Sustainable cities and communities — Vocabulary". 2016.

[3] buildingSMART International. "What is openBIM?". 2020. https://www.buildingsmart.org/about/openbim/openbim-definition/ (accessed Jul. 17, 2023).

[4] buildingSMART International. "Industry Foundation Classes (IFC)". 2019. https://technical.buildingsmart.org/standards/ifc/ (accessed Jul. 15, 2023).

[5] Janssen, P., Chen, K. W., Mohanty, A. "Automated Generation of BIM Models". *Proc. Int. Conf. Educ. Res. Comput. Aided Archit. Des. Eur.* 2016. Vol. 2. pp. 583–590. DOI: 10.52842/conf.ecaade.2016.2.583

[6] Chi, H. L., Wang, X., Jiao, Y. "BIM-Enabled Structural Design: Impacts and Future Developments in Structural Modelling, Analysis and Optimisation Processes". *Arch. Comput. Methods Eng.* 2015. Vol. 22. No. 1. pp. 135–151. DOI: 10.1007/s11831-014-9127-7

[7] Lee, P. C., Lo, T. P., Tian, M. Y., Long, D. "An Efficient Design Support System based on Automatic Rule Checking and Case-based Reasoning". *KSCE J. Civ. Eng.* 2019. Vol. 23. pp. 1952–1962, 2019. DOI: 10.1007/s12205-019-1750-2

[8] Ghannad, P., Lee, Y. C., Dimyadi, J., Solihin, W. "Automated BIM data validation integrating open-standard schema with visual programming language". *Adv. Eng. Informatics*. March 2019. Vol. 40. pp. 14–28. DOI: 10.1016/j.aei.2019.01.006

[9] Sena, P. C. P. de. "Automação de processos de projeto e programação em BIM: Dynamo, Python e C#". 2019. Universidade de São Paulo.

[10] Leger, S. "Blender-Archipack". 2020. https://blender-archipack.org/es/ (accessed Mar. 01, 2023).

[11] Wiki.OSArch. "BlenderBIM Add-on". 2022. https://wiki.osarch.org/index.php?title=BlenderBIM_Add-on (accessed Oct. 20, 2022).

[12] IfcOpenShell Contributors. "BlenderBIM Add-on documentation". 2022. https://blenderbim.org/docs/index.html (accessed Mar. 01, 2023).

[13] Tobiáš, P. "An Investigation into the Possibilities of BIM and GIS Cooperation and Utilization of GIS in the BIM Process". *Geoinformatics FCE CTU*. 2015. Vol. 14. No. 1. pp. 65–78. DOI: 10.14311/gi.14.1.5

[14] buildingSMART International. “IFC Version 4.0 - Addendum 2 - Technical Corrigendum 1”. 2017. https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/ (accessed Jul. 15, 2023).
[15] Delft University of Technology. “GeoBIM benchmark - IFC”. 2019. https://3d.bk.tudelft.nl/projects/geobim-benchmark/ifc.html (accessed Jul. 15, 2023).
[16] Borrmann, A., Beetz, J., Koch, C., Liebich, T., Muhic, S. “Industry foundation classes: A standardized data model for the vendor-neutral exchange of digital building models”. 2018. DOI: 10.1007/978-3-319-92862-3_5
[17] Donkers, S. “Automatic generation of CityGML LoD3 building models from IFC models”. 2013. Delft University of Technology. https://repository.tudelft.nl/islandora/object/uuid%3A31380219-f8e8-4c66-a2dc-548c3680bb8d (accessed Jul. 15, 2023).

ACKNOWLEDGEMENTS

This research has received co-funding support by the Agencia Canaria de Investigación, Innovación y Sociedad de la Información (ACIISI) de la Consejería de Economía, Conocimiento y Empleo; and by the European Social Fund (ESF), Canary Islands Integrated Operational Programme 2014-2020, Axis 3, Priority Theme 74 (85%).

SUPPLEMENTARY MATERIAL

Link to the GitHub repository: https://github.com/TechnoLabULL/IfcClassAssignment