IFC2BCM
A Tool for Generating IndoorGML and Building Configuration Model from IFC

Jia, Zhuoran; Nourian, Pirouz; Luscuere, Peter; Wagenaar, Cor

DOI
10.1016/j.softx.2024.101975

Publication date
2025

Document Version
Final published version

Published in
SoftwareX

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

IFC2BCM: A Tool for Generating IndoorGML and Building Configuration Model from IFC

Zhuroan Jia [a,*], Pirouz Nourian [b], Peter Luscuere [a], Cor Wagenaar [a,c]

[a] *Faculty of Architecture and the Built Environment, Delft University of Technology, Julianalaan 134, Delft, 2628 BL, The Netherlands*
[b] *Faculty of Geo-Information Science and Earth Observation, University of Twente, DRIENERLOLAAN 5 7522 NB ENSCHEDE, The Netherlands*
[c] *History of Architecture and Urbanism – Faculty Board, Faculty of Arts, University of Groningen, Oude Boteringestraat 34, Groningen, 9712 GK, The Netherlands*

ARTICLE INFO

Keywords:
Digital twin
Building configuration model
Decision support system
IndoorGML

ABSTRACT

IFC2BCM is a novel software tool designed to generate IndoorGML and Building Configuration Models (BCM) from IFC/BIM models. The primary motivation behind IFC2BCM is to develop a tool for generating BCM as the core foundation of a Spatial Design Support System that will evaluate layout designs of complex buildings such as hospitals regarding operational efficiency. The software addresses the need for detailed spatial network analysis and simulation modelling in complex environments, offering a semi-automatic process to convert IFC data into IndoorGML, and subsequently into a comprehensive BCM. The BCM generated by this tool consists of geometric, topological, semantic, and operational information, it supports applications such as space optimization, facility management, ensuring safety, and indoor navigation. More generally, the results are relevant to the study of complex buildings such as airports, transport hubs, public buildings, etc.

Code metadata

Current code version	v1.1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00343
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/8363427/tree
Legal Code License	MIT License
Code versioning system used	git
Software code languages, tools, and services used	Python, Jupyter Notebook, Autodesk's Revit and Dynamo, McNeel's Rhino and Grasshopper
Compilation requirements, operating environments & dependencies	dependencies for python: pandas, NumPy, COMPAS, Matplotlib, NetworkX, LXML. dependencies for Grasshopper: Human, LunchBox, LunchBoxML.
If available Link to developer documentation/manual	None
Support email for questions	Z.Jia@tudelft.nl

1. Introduction

1.1. Motivation and significance

The spatial configuration of complex buildings significantly impacts their functionality, creating varying distances and connections. Unlike traditional architectural analysis, configurational design focuses on the spaces within a building rather than its physical boundaries. It is challenging to obtain an explicit model of these internal spaces using standard Building Information Models (BIM). This paper introduces a digital workflow to extract a Building Configuration Information Model (BCM) from BIM models in Industry Foundation Classes (IFC) file formats. We designed this workflow for general use in supporting the design and analysis of different types of complex buildings such as airports, transport hubs, museums, hospitals, etc. In this study, among all types of complex buildings, we chose hospitals as the case, because the hospital is a very representative type of complex buildings. Its complexities are twofold, firstly, the spatial complexity of a hospital can be compared to small cities where corridors in hospitals are similar

* Corresponding author.
E-mail address: Z.Jia@tudelft.nl (Z. Jia).

https://doi.org/10.1016/j.softx.2024.101975
Received 27 June 2024; Received in revised form 6 November 2024; Accepted 8 November 2024
Available online 1 December 2024

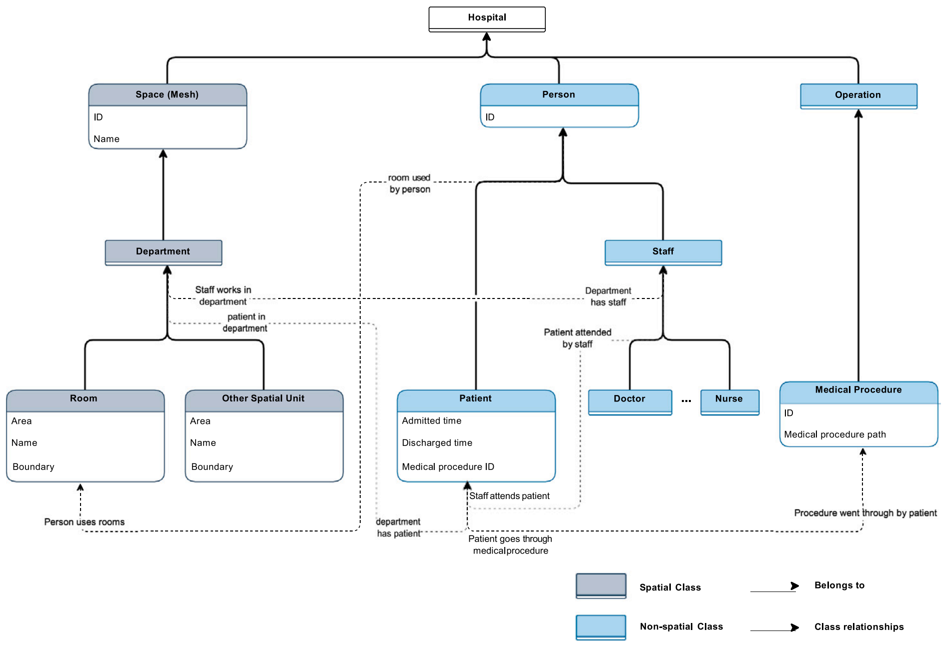


Fig. 1. An UML diagram illustrating data included in a Hospital Configuration Model [3].

to roads in a city and different rooms with various functions in a hospital are similar to different land uses in a city [1]. Secondly, the procedural/operational complexity in a hospital is significant. Hospitals function as a 'healing factory' where multiple procedures (e.g., diagnostic procedures, surgical procedures, emergency and critical care procedures, etc.) take place simultaneously [1]. Hence in this paper, our attention is focused on hospitals.

Our research is part of a project developing Hospital Design Support Systems (HDSS), which incorporate early operational insights to improve hospital layouts. The HDSS uses spatial analysis and simulation modelling to evaluate hospital efficiency, requiring a Hospital Configuration Model (HCM) that includes spatial and non-spatial information. Spatial information encompasses geometric and topological data, while non-spatial information includes semantic and operational details. For instance, geometric data can be room boundaries defined by lists of vertices with 3D coordinates, and topological data can be a graph that illustrates relationships between spatial units. Semantic information might include room names and areas, while operational information covers medical processes in hospitals such as patient journeys (see Fig. 6). Fig. 1 illustrates the spatial and non-spatial information in an HCM and the relationship between different types of information. Consistency among these data types ensures effective operational management in hospitals.

According to our literature study (see Section 1.4), there is no available tool that can generate hospital configuration models or building configuration models. To address the lack of tools for generating building configuration models, we developed IFC2BCM [2], a tool for semi-automatically creating BCM/HCM from BIM/IFC models. This tool forms the core of the HDSS, enabling the evaluation of hospital layout designs in terms of functionality and efficiency.

1.2. Contribution

The contributions of this software are summarized as follows:

- *IFC2BCM* can semi-automatically generate correct IndoorGML file which is Open Geospatial Consortium (OGC) standard for representing and exchanging indoor spatial information [4]. IndoorGML consists of four groups of spatial data, namely, CellSpace, CellSpaceBoundary, Node, and Edge. The term 'CellSpace' refers to the room or corridor in a building, 'CellSpaceBoundary' refers to the door, 'Node' is a point representation of the room, and an 'Edge' connects two nodes if the two corresponding rooms of the nodes are adjacent. IndoorGML files can facilitate applications in indoor navigation and facility management [5]. However, there is a lack of available IndoorGML files, and also a lack of appropriate tools for correctly generating IndoorGML files. Our model resolves these limitations, as it contributes to providing more IndoorGML files with correct structures and necessary information. It also works for any type of building input (e.g., buildings with regular/irregular shapes, or buildings with simple/complex indoor space, etc.).
- Although IndoorGML is designed to support applications in indoor navigation and facility management, to be able to accomplish such tasks, IndoorGML models often need to be equipped with other data such as operational information and meaningful semantic information. For instance, if we want to simulate the operations in a hospital using a hospital IndoorGML model, besides the geometric and topological information that the hospital IndoorGML has, we also need to acquire the operational information of the hospital (e.g., the patient journey in hospital) and semantic information related to hospital organizations. IndoorGML files are facing the challenge of missing such information. The HCM generated by our software addresses this challenge. In this study, we developed software functions for extracting hierarchical semantic information according to hospital organizational structures into HCM. We also developed functions for extracting operational information (i.e., patient journey in the form of Python lists) from available data (Fig. 6) into HCM.
- Another challenge that IndoorGML faces is that it is encoded in XML (eXtensible Markup Language) format [6], which is tedious, deeply hierarchical, complicated and not well-suited for the web [7]. These features of XML encoding make IndoorGML files very hard to parse and collect information from. As a result, there is a limited number of software packages supporting IndoorGML, and a limited number of available IndoorGML files [7]. By contrast, our BCM/HCM files are encoded in JSON (JavaScript Object Notation) format [8], which is a more popular exchange format with more available libraries and users. We use JSON format to encode BCM/HCM in a more 'flattened out' structure [7], which

makes the BCM/HCM more editable and easier to understand by humans compared to IndoorGML.

- The BCM generated by *IFC2BCM* can support multiple research applications such as space optimization, facility management, indoor navigation, wayfinding, etc.

Jia et al. [3] used this software to develop an HCM as the core of HDSS for assessing hospital layouts' efficiencies and efficacy in terms of four performance indicators, i.e., crowdedness in hospital space, patient waiting time, patient walking distance, and difficulty in way-finding).

1.3. Experimental setting

IFC2BCM is designed for semi-automatically converting BIM/IFC files into IndoorGML and generating BCM/HCM from IndoorGML. The software used for developing *IFC2BCM* includes Autodesk's Revit and Dynamo, McNeel's Rhino and Grasshopper, and Python. Dependencies are also needed for this tool. The dependencies for Grasshopper include Human, LunchBox, and LunchBoxML. The libraries for Python are pandas, NumPy, COMPAS, Matplotlib, NetworkX, and LXML. The data used for this experiment is an open-source hospital IFC file [9].

The experimental procedure includes three main steps, which are summarized as follows:

- **Step 1, Importing IFC.** Open the open-source IFC file with Autodesk's Revit. In Revit, open the Dynamo file 'Home.dyn' which is located in 'geometry_software' directory of the repository. This step will produce two output files, one is for the building's room boundaries, and another is for the building's door locations.
- **Step 2, Generating IndoorGML.** This step can be subdivided into five sub-steps.
  Sub-step 1, **generating CellSpace for IndoorGML**. Put the room boundary output file from step 1 into the 'edit_csv.ipynb' file located in the 'notes/csv processor' directory of the repository, and process the boundary file, the processed boundary here is the room lower boundary (i.e., floor boundary), then put the processed file into the 'add_storey.ipynb' file to get room storeys. The functions in the 'add_storey.ipynb' file assign each room a storey. Each room's Z coordinate was checked, if it is zero, the functions assign level 0 to this room, if it is 4.57, the functions assign level 1 to this room, and if the Z coordinate is 9.25, the functions assign level 2 to this room. Next, put the room lower boundary file into the 'create_upper_boundary.ipynb' file to get the room upper boundary (ceiling boundary) file, here room boundaries' Z coordinates were changed according to the storey. Then put both room lower boundary and upper boundary files into 'create_wall_boundary.ipynb' to get the wall boundary file. Lastly, put room lower boundary, upper boundary, and wall boundary files into 'add_all_csBoundary_together.ipynb' to create the CellSpace data.
  Sub-step 2, **generating CellSpaceBoundary for IndoorGML**. Put the door location output file from step 1 into 'edit_door_csv.ipynb' located in the 'notes/csv processor' directory of the repository for processing the data and make it readable by Grasshopper, then put the processed data into the Grasshopper file named 'create_graph.gh' in 'geometry_software' directory of the repository, and create door boundaries. The Grasshopper scripts treat the door's location as a lower centre point and draw the door boundary upwards. Next, put the door boundaries file into 'edit_door_boundaries.ipynb' to get the final CellSpaceBoundary data.
  Sub-step 3, **generating nodes for IndoorGML**. In the room boundary file generated by step 1, select only corridor boundaries to obtain a corridor boundary file, and select only stair boundaries to obtain a stair boundary file. Then put the room boundary file, corridor boundary file, and stair boundary file into the 'edit_csv.ipynb' file to process them and make them readable by Grasshopper. Then put the processed files into the Grasshopper file 'create_graph.gh' to get nodes and edges data. The nodes were created by finding each room boundary's centre point, and the edges were created by connecting the room boundary's centre point to its corresponding door location point. Specifically, the algorithms in the Grasshopper check if the door location point is on the room boundary, if yes, connect the door location point to this room boundary's centre point, if not, skip. Subsequently, put the nodes and edges data into the 'add_edges_to_nodes.ipynb' located in the 'notes/csv processor' directory of the repository to add both groups of data together to obtain the final nodes file.
  Sub-step 4, **generating edges for IndoorGML**. In the last sub-step, from the Grasshopper file 'create_graph.gh', export all edges' start and end points to a csv file, and then put this output file together with the final nodes file from the last sub-step into 'create_transitions.ipynb' in 'notes/csv processor' directory of the repository to generate final edge data.
  Sub-step 5, **generating IndoorGML**. Put all four final output files from previous sub-steps into 'etree_to_gml.ipynb' in the 'IndoorGML Generator' directory of the repository to obtain the IndoorGML file. This Python generator uses LXML's etree module [10] for encoding XML files. The functions were designed according to the XML's structure for creating properly structured IndoorGML files.
- **Step 3, Generating HCM.** Import the IndoorGML file from step 2 into the IndoorGML parser named 'ig2ij.ipynb' in the 'HCM generator' in the 'notes/HCM generator' directory of the repository to get a JSON [8] file, which is encoded in a more 'flattened out' and editable structure [7]. Then put the JSON file into 'ij2cp_and_nx.ipynb' for extracting geometric and topological information for the HCM. The geometric information was extracted into COMPAS mesh and visualized using COMPAS library [11], and the topological information was extracted into a network graph using NetworkX [12]. The JSON file can also be put into the 'ij2semantic_and_operational_info.ipynb' file for extracting semantic and operational information. The semantic information was extracted into a Python dictionary demonstrating hospital departments and all the rooms within their respective departments. The operational information is extracted into Python lists indicating patient journeys in the hospital. All these four types of information constitute the HCM.

1.4. Related works

Our work is inspired by Tong and Zheng's work [13]. They developed a tool for transforming IFC models to IndoorGML files using Autodesk Revit and Dynamo, McNeel Rhino and Grasshopper, and Python. However, the generated IndoorGML files are not equivalent to configuration models, and this tool is limited to modular buildings with simple geometries, such as rectangular rooms with four sides. It does not work for buildings with complex shapes. Our software is built on Tong and Zheng's tool, and is equipped with the function of generating HCM from an IndoorGML file, it also offers a more generalized solution that works for buildings with rooms of irregular shapes. Other software used in our study includes an IndoorGML parser developed by Ledoux [14].

Another related work is a tool developed by Diakite et al. [5], they created a C++ tool that automatically generates IndoorGML models from IFC models, but the resulting IndoorGML files lack semantic information.

Intratech [15] developed a plugin for AutoCAD and Revit to extract IndoorGML. However, this plugin also generates IndoorGMLs that lack semantic information, and this plugin relies on exclusive formats.

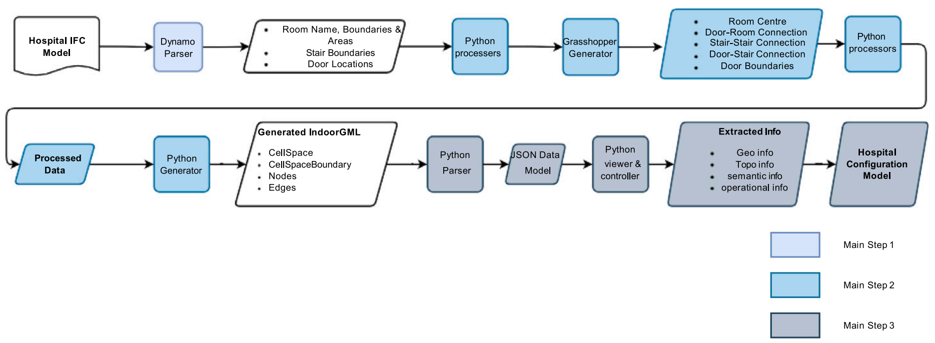


Fig. 2. The workflow of the software system [3].

2. Software description

2.1. Software architecture

Fig. 2 illustrates the detailed workflow of the software system. As described in Section 1.3, the workflow includes three main steps. The first step is to import the IFC file into the Dynamo parser to extract relevant information.

The second step is to put the extracted data into Python processors as well as Grasshopper and Python generators for generating IndoorGML files. Because Dynamo and Grasshopper utilize different data structures, the data exported from Dynamo in Step 1 needs to be first processed in Python processors to be compatible with Grasshopper generators. Once the data is processed and imported into the Grasshopper generator, the scripts inside the Grasshopper generator read the data and generate the necessary components for the IndoorGML model. The data made by the Grasshopper generator again needs to be processed in Python processors to become compatible with the Python generator that will write the final IndoorGML file. Given that the IndoorGML file uses an XML-based exchange format [16], we use etree [10], an XML library for Python, for scripting our Python generator.

The last step of the workflow is to create an HCM file from the IndoorGML file. The IndoorGML model was first parsed into a JSON file by the Python parser that was developed by Ledoux [14]. The JSON file was then processed by the Python viewer and controller for extracting semantic and operational information and integrating them with geometric and topological information to form the HCM. Libraries used for scripting the Python viewer and controller include COMPAS [11], NetworkX [12], and matplotlib [17].

2.2. Software functionalities

The major functionalities of the software are summarized as follows:

- Major function 1, Semi-automatically producing IndoorGML files with semantic information that can be applied in indoor navigation and facility management. Compared to Tong and Zheng's software [13], our software's functionality is more generalized and works for complex-shaped buildings.
- Major function 2, Automatically generating BCM/HCM containing four types of information (i.e., geometric information, topological information, semantic information, and operational information) from IndoorGML file, which can be used for spatial network analysis and simulation modelling. This major function is composed of four minor functions. The first minor function is to extract the geometry of the interior space of the IndoorGML model and convert it into a mesh and visualize it. The second minor function is to extract the topological information of the IndoorGML model and convert it into a graph and visualize it. The third minor function is to extract the semantic information of the IndoorGML model and convert it into a Python dictionary with a hierarchical structure. The fourth minor function is to extract operational information (i.e., patient journey data) from available documents related to hospital procedures and convert such information into Python lists for further simulation uses.

3. Illustrative examples

This section shows an instance of HCM, and it shows what exactly it contains. We used a real-world hospital's IFC model [9] as our input and used our software to convert it into an HCM. Fig. 3 illustrates the geometric and topological information of the HCM, where the red graph is embedded in the transparent building geometry. It is to be noticed that for clarity reasons, we only visualized one floor of the hospital building instead of all three floors. Fig. 4 shows the hierarchical semantic information of the HCM as well as the function codes of how to extract such information and organize it into a hierarchical structure. Table 2 is the resulting Python dictionary of the extracted departments and their rooms. Table 1's right column shows HCM's operational information, since the hospital IFC model does not contain operational information such as medical procedures, we need to extract such information from other sources. We selected representative hospital operational information pertaining to patient journeys from Peng's study [18] and reproduced this information in the form of a BPMN flow chart (see Fig. 6). BPMN is an industry-standard using flow charts to illustrate system processes [19]. Fig. 6 illustrates typical patient journeys in the outpatient department. We converted the patient journeys in this figure into Python lists as shown in the left column of Table 1. In these lists, each element is a place in the hospital that the patient needs to go, and the entire list is the patient's journey. What we did next was to use these lists as inputs for our tool to generate the operational information. Specifically, for each element (i.e., space) in the input list, we used our tool to find its corresponding room names from the HCM's semantic information which contains all the room names of the hospital, and put all corresponding room names into a new list to form the patient path data (right column of Table 1). This patient path data is HCM's operational information and will be used for HDSS's simulation modelling process. It is to be noticed that the element in the input data list might have multiple corresponding nodes, for example, there are multiple 'registration stations' or 'waiting areas' in our hospital case, and we only chose the appropriate ones to form the output data list according to the IFC model.

Fig. 5 proves our software's ability of handling complex building geometries. This figure is the second-floor plan of the selected hospital case study, and it shows the geometric complexities of the hospital space. Several corridors with irregular shapes (e.g., multiple turns and corners) are highlighted in red. These irregular-shaped corridors were successfully handled by our tool for generating correct IndoorGML files. By contrast, Tong and Zheng's tool [13] failed to generate the correct IndoorGML file for this hospital case.

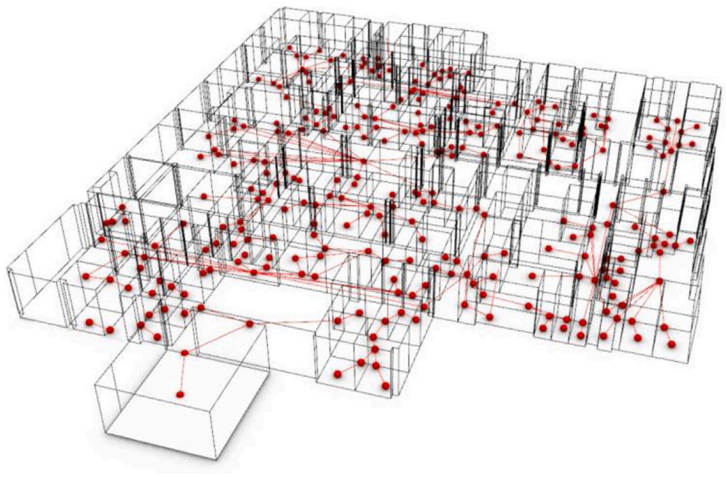
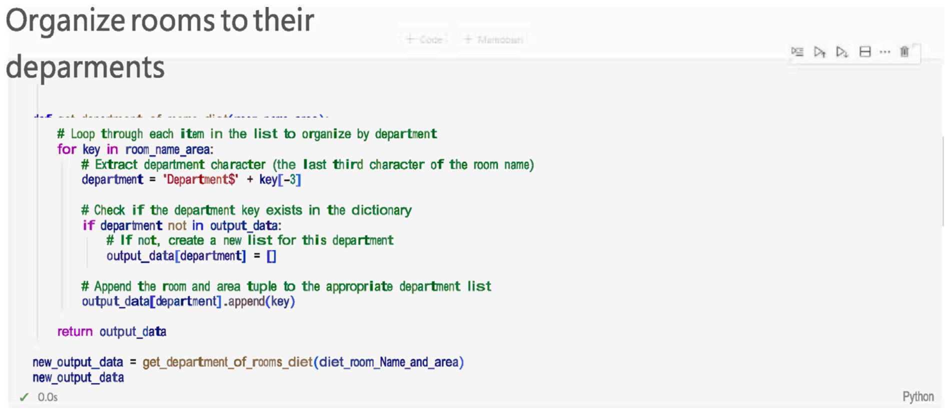


Fig. 3. Visualization of an HCM's geometric and topological information.



Organize rooms to their deparments

```
    # Loop through each item in the list to organize by department
    for key in room_name_area:
        # Extract department character (the last third character of the room name)
        department = 'Department$' + key[-3]

        # Check if the department key exists in the dictionary
        if department not in output_data:
            # If not, create a new list for this department
            output_data[department] = []

        # Append the room and area tuple to the appropriate department list
        output_data[department].append(key)

    return output_data

new_output_data = get_department_of_rooms_diet(diet_room_Name_and_area)
new_output_data
```

✓ 0.0s

Python

Fig. 4. An HCM's semantic information.

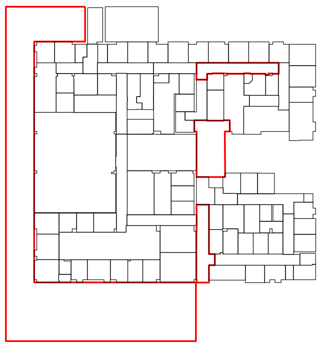


Fig. 5. Second Floor Plan of the selected hospital case.

4. Impact

4.1. Contributions to future research

One of the potential Future research directions is to use the BCM as an input for developing Spatial Decision Support Systems (e.g., HDSS), which applies methods of spatial network analysis and simulation modelling for evaluating service accessibility and mobility efficiency in complex buildings such as hospitals, airports, and transport hubs, etc.

Another possible direction for future research is to develop a standardized method for automatically extracting hospital operational data, such as medical procedures, into a data model utilizing business process model notation (BPMN) [19] or enterprise resource planning (ERP) system [20] techniques. BPMN is an industry standard for business process modelling, utilizing flow charts to depict the steps involved in a business process [19].

It is straightforward to understand that the necessary operational information for comprehending hospital procedures can be extracted into BPMN diagrams. Thus, creating a systematic approach for the automatic extraction of such information can be a desirable future research direction. We propose that an expert, such as an Industrial Engineer or someone knowledgeable in Operations Research, should systematically extract this information from textual and visual documents related to the operational management and service design of a hospital to build BPMN models for describing the main procedural workflows in the hospital. These models, providing operational information, can be integrated into the HCM. In Fig. 6, we illustrate how the BPMN model of the operational information in a real-world hospital should look.

4.2. Contributions to current research

The ways how our software improves the pursuit of existing research are summarized as follows:

- *IFC2BCM* can semi-automatically generate a BCM which can facilitate space optimization by serving as a foundation to analyse the relationships and flows between various spatial units.
- With operational information, an HCM generated by *IFC2BCM* can serve as the basis of a digital twin for simulating and monitoring the medical processes taking place in a hospital, such as operational management and facility management.

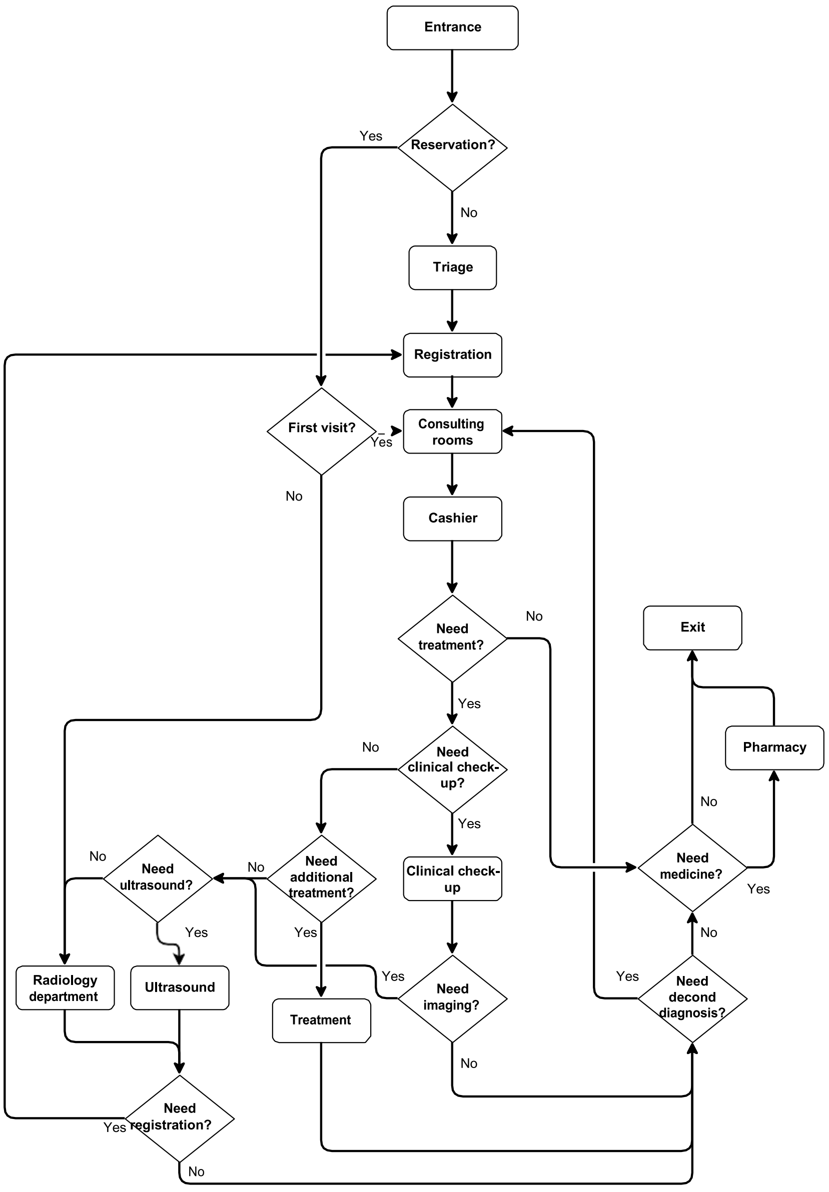


Fig. 6. Patients' Paths in Outpatient Department of Panyu Central Hospital.
Source: image source: [3,18].

- A BCM generated by *IFC2BCM* can enhance a building's safety by strategically positioning guards or cameras to achieve optimal coverage with the minimal necessary number of guards or cameras.
- After the building is constructed, we can use *IFC2BCM* to generate the building's BCM and augment it with 3D information to create a model for indoor navigation and way-finding.

4.3. Impact pathway

This subsection introduces the impact pathway of IFC2BCM. Impact pathway is a concept model proposed by the Dutch Research Council (NWO) [21]. It outlines the process through which a research project's outputs lead to intended outcomes, and result in some impacts, where outputs are direct findings of the research project, outcomes are changes in stakeholders' behaviours and activities due to the application of outputs, and impacts are changes in economic, environmental or social conditions caused by outputs [21]. Fig. 7 illustrates the impact pathway of our software. The direct outputs of our software are generated IndoorGML and BCM/HCM models. These outputs can lead to the intermediate outcomes that more researchers will use this software to generate IndoorGMLs and BCMs for supporting applications such as space optimization, facility management, indoor safety improvement, and indoor navigation. These intermediate outcomes, together with the future work of the development of an HDSS, can further contribute to the outcomes that architects and hospital directors design better hospitals in terms of operational efficiency, which ultimately leads to the impacts of reduced hospital expenditures and improved public health.

4.4. Application

Jia et al. [3] used this software in a study for constructing a hospital configuration model as the core of a Hospital Design Support System (HDSS) for assessing the performances of hospital layout designs in terms of crowdingness in hospital spaces, patient waiting time, patient walking distance, and difficulty in way-finding. In this study, we first stated that it is beneficial to use spatial design support systems such as HDSS for assessing hospital performances in terms of accessibility and

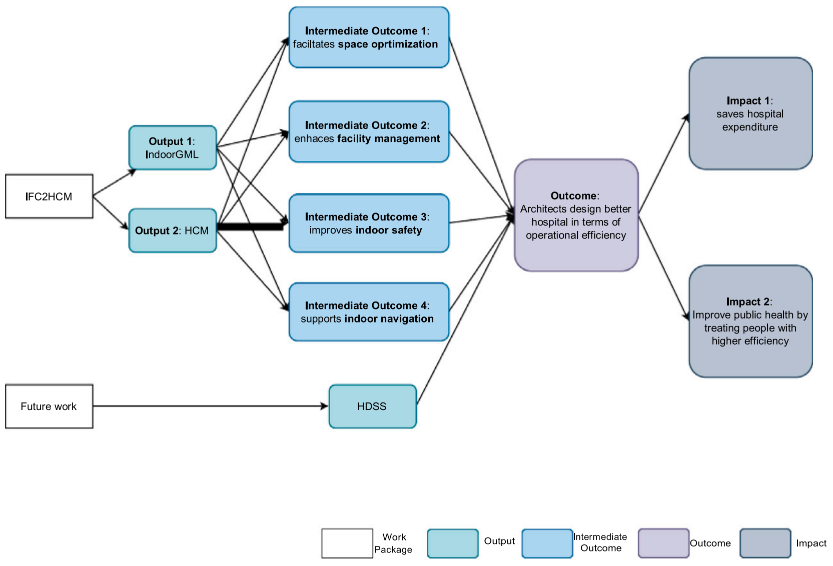

Intermediate Outcome 1: facilitates space oprtimization
Intermediate Outcome 2: enhaces facility management
Intermediate Outcome 3: improves indoor safety
Intermediate Outcome 4: supports indoor navigation
Output 1: IndoorGML
Output 2: HCM
IFC2HCM
Future work
HDSS
Outcome: Architects design better hospital in terms of operational efficiency
Impact 1: saves hospital expenditure
Impact 2: Improve public health by treating people with higher efficiency
Work Package
Output
Intermediate Outcome
Outcome
Impact

Fig. 7. Impact pathway of IFC2BCM.

Table 1
Input and Output data list of HCM's operational information (notice: for simplicity reasons not all data are shown in this table, for complete data please go to https://github.com/ZhuoranJia/IFC2BCM).

Input data list	Output data list
origianl_medical_path_1 = [‘registration’, ‘triage’, ‘waiting’, ‘diagnosis’, ‘medicine’]	medical_path_1 = [‘RECEPTION1B13’, ‘WTSandMEAS.ROOM1D15’, ‘WTSandMEAS.ROOM1D30’, WAITING/ACTIVITYAREA1DC1’, ‘INTERACTIONSTATION1D11’, …, ‘PHARM.DISP.1A16’]
origianl_medical_path_2 = [‘registration’, ‘triage’, ‘waiting’, ‘diagnosis’, ‘waiting’, ‘clinical-checkups’, ‘medicine’]	medical_path_2 = [‘RECEPTION1B13’, ‘WTSandMEAS.ROOM1D15’, ‘WTSandMEAS.ROOM1D30’, ‘WAITING/ACTIVITYAREA1DC1’, ‘INTERACTIONSTATION1D11’, …, ‘CENTRALWAITING1AC1’, ’BLOODDRAW1B03’, ‘PHARM.DISP.1A16’]
origianl_medical_path_3 = [‘registration’, ‘triage’, ‘waiting’, ‘diganosis’, ‘waiting’, ‘imaging’, ‘medicine’]	medical_path_3 = [‘RECEPTION1B13’, ‘WTSandMEAS.ROOM1D15’, ‘WTSandMEAS.ROOM1D30’, ‘WAITING/ACTIVITYAREA1DC1’, ‘INTERACTIONSTATION1D11’, …, ‘CENTRALWAITING1AC1’, ‘RADIOGRAPHICROOM1B19’, ‘PHARM.DISP.1A16’]

Table 2
Extracted semantic information of the HCM (notice: for simplicity reasons not all data are shown in this table, for complete data please go to https://github.com/ZhuoranJia/IFC2BCM).

Departments & Rooms
{Department$A’: [‘CENTRALWAITING1AC1’, ‘CORRIDOR2AC3’, ‘PHARM.DISP.1A16’, ‘CORRIDOR2AC1’, ‘DENTALWAITING2A11’, … ‘X-RAYALCOVE2A12-A’]}
{Department$B’: [‘CORRIDOR1BC2’, ‘LAB1B04’, ‘CORRIDOR1BC4’, … ‘RECEPTION1B01’, ‘RECEPTION1B13’, ‘TECHOFFICE2B9’]}
{Department$D’: [‘WAITING/ACTIVITYAREA1DC1’, ‘MAINMECHANICALROOM2D05’, … ‘INTERACTIONSTATION1D11’, ‘INTERACTIONSTATION1D07’, ‘INTERACTIONSTATION1D08’, ‘INTERACTIONSTATION1D09’, ‘INTERACTIONSTATION1D28’, ‘INTERACTIONSTATION1D34’, ‘INTERACTIONSTATION1D35’, … ‘COMPUTERROOM2D04A’]}
…

mobility because such systems can provide intuitive and explainable assessment mechanisms for design decision support. We then argued that the HCM is the core of HDSS because it will be what we evaluate. Specifically, the HCM contains four types of information, i.e., geometric information, topological information, semantic information, and operational information. These types of information will be the inputs for running the assessments.

To determine what specific data of each type is needed in the HCM, We first envisaged the use cases for the HDSS, then based on the use cases, we decide what specific data we want in the HCM. For example, one of the use cases is that the architect can use this HDSS to check how long a patient needs to walk in the hospital to complete this patient's medical procedure. In this use case, the operational information of the patient's medical procedure is needed because from the patient medical procedure, we can extract the patient's path which shows all the rooms the patient needs to go to in order to complete the medical procedure. The topological information needed in this use case is a network graph of the hospital layout. The network graph contains nodes and edges where nodes represent spatial units in the hospital and edges show their relationships. We also need to add semantic information to the graph, i.e., we need to add each spatial unit's name to its corresponding node, so that we can find the specific patient path in the graph. Lastly, we need to add geometric information to the graph, in other words, we need to add each node's 3D coordinates so that we will be able to calculate the distance of this patient path. For other use cases and other specific data needed in the HCM please refer to [3].

5. Conclusions

We developed a tool for converting IFC/BIM models into IndoorGML and subsequently into Building Configuration Models (BCMs), this tool

Table 3
Differences between IndoorGML and BCM.

	IndoorGML	BCM
Information Content	- Geometric Info - Topological Info - Some IndoorGML files contain unstructured semantic info, while others do not	- Geometric Info - Topological Info - Hierarchical semantic info which facilitates simulation modelling - Operational Info which facilitates simulation modelling
Encoding	XML	JSON
edit-ability	Low edit-ability: difficult to add/remove contents to/from IndoorGML	High edit-ability: easy to add/remove contents to/from BCM

provides the basis for developing the Spatial Design Support System, which uses methods of spatial analysis and simulation modelling for evaluating service accessibility and mobility efficiency in complex buildings. Addressing the lack of tools for generating IndoorGMLs and enhancing IndoorGML files with operational and semantic data, IFC2BCM uses JSON encoding to improve accessibility and usability.

Table 3 summarizes the differences between IndoorGML files and BCMs. This software's robustness and flexibility make it applicable to various building types, including hospitals, airports, and transport hubs, highlighting its broader relevance and potential impact on the design and operational efficiency of complex environments. However, this software still has several limitations which are summarized as follows:

- **User Experience:** This software is written in Python scripts, Grasshopper scripts, and Dynamo scripts. Users are required to switch between these three tools to operate the software, which significantly complicates the user experience.
- **Software's Accuracy:** When creating the graph for the hospital layout, this software generates edges by connecting the room's node to its corresponding door's node. This means the software will only connect two spatial units if they are connected by a door. For example, if a room and a corridor are connected to the same door, then this room and this corridor will be connected through that door. If one corridor is directly connected to another corridor and there is no door connecting them, the software will not link these two corridors and leave them disconnected, which is inaccurate because if two spatial units are directly connected without doors, they should be linked by an edge in the graph.
- Time Efficiency: The software's overall time efficiency performance is adequate. However, the scripts for visualizing the mesh geometry of the HCM is time-consuming. As the input IFC model gets bigger, more time will be taken to visualize the model's geometry.

These limitations lead to the future works:

- Integrating the scripts in different tools into one for a simpler user experience.
- Improving software's accuracy in terms of generating edges of hospital's layout graph.
- Improving software's function of visualizing geometric information of the HCM so it can be less time-consuming.

Other future works will include developing the Hospital Design Support System and enhancing operational data extraction.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

CRediT authorship contribution statement

Zhuroan Jia: Writing – original draft, Software, Methodology, Data curation, Conceptualization. **Pirouz Nourian:** Writing – review & editing, Supervision, Software, Methodology, Conceptualization. **Peter Luscuere:** Supervision, Project administration. **Cor Wagenaar:** Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Jia Zhuoran, Nourian Pirouz, Luscuere Peter, Wagenaar Cor. Spatial decision support systems for hospital layout design: A review. J Build Eng 2023;67(106042). http://dx.doi.org/10.1016/j.jobe.2023.106042, url: http://www.scopus.com/inward/record.url?scp=85147548991&partnerID=8YFLogxK.

[2] Jia Zhuoran, Nourian Pirouz. IFC2BCM. Zenodo; 2024, http://dx.doi.org/10.5281/zenodo.11466047.

[3] Jia Zhuoran, Nourian Pirouz, Luscuere Peter, Wagenaar Cor. A configuration model for hospital design support systems. 2024, http://dx.doi.org/10.20944/preprints202411.2191.v1, url: https://www.preprints.org/manuscript/202411.2191/v1.

[4] OGC® IndoorGML 1.1, url: https://docs.ogc.org/is/19-011r4/19-011r4.html.

[5] Diakite AA, Díaz-Vilariño L, Biljecki F, Isikdag Ü, Simmons S, Li K, et al. IFC2IndoorGML: An open-source tool for generating IndoorGML from IFC. Int Arch Photogramm Remote Sens Spatial Inf Sci 2022;XLIII-B4-2022:295–301. http://dx.doi.org/10.5194/isprs-archives-XLIII-B4-2022-295-2022, Conference Name: XXIV ISPRS Congress "Imaging today, foreseeing tomorrow", Commission IV - 2022 edition, 6&ndash;11 June 2022, Nice, France Publisher: Copernicus GmbH, url: https://isprs-archives.copernicus.org/articles/XLIII-B4-2022/295/2022/.

[6] W3C Recommendation. Extensible Markup Language (XML) 1.0 (fifth edition). 2008, url: https://www.w3.org/TR/xml/#sec-intro.

[7] Ohori Ken Arroyo, Ledoux Hugo, Peters Ravi. 3D modelling of the built environment. 2022, url: https://3d.bk.tudelft.nl/courses/geo1004/data/3dbook.pdf.

[8] Python Software Foundation. json — JSON encoder and decoder, url: https://docs.python.org/3/library/json.html.

[9] van Berlo Léon. Sample-Test-Files/IFC 2x3/Medical-Dental Clinic at master · buildingSMART/Sample-Test-Files · GitHub, url: https://github.com/buildingSMART/Sample-Test-Files/tree/master/IFC%202x3/Medical-Dental%20Clinic.

[10] Python Software Foundation. xml.etree.ElementTree — The ElementTree XML API. 2024, url: https://docs.python.org/3/library/xml.etree.elementtree.html.

[11] Van Mele Tom, et al. COMPAS: A framework for computational research in architecture and structures. 2017, http://dx.doi.org/10.5281/zenodo.2594510.

[12] Hagberg Aric A, Schult Daniel A, Swart Pieter J. Exploring network structure, dynamics, and function using NetworkX. In: Proceedings of the 7th python in science conference. 2008, p. 11–5.

[13] Tong Ziyu, Zheng Hang. Generation scheme of IndoorGML model based on building information model. In: Yan Chao, Chai Hua, Sun Tongyue, Yuan Philip F, editors. Phygital intelligence. Singapore: Springer Nature; 2024, p. 225–34. http://dx.doi.org/10.1007/978-981-99-8405-3_19.

[14] Ledoux Hugo. indoorjson/software/ig2ij at master · tudelft3d/indoorjson · GitHub. 2020, url: https://github.com/tudelft3d/indoorjson/tree/master/software/ig2ij.

[15] Intratech. intratech/KICT-Autocad-RevitToIndoorGML. Intratech; 2022, original-date: 2020-11-24T01:47:31Z, url: https://github.com/intratech/KICT-Autocad-RevitToIndoorGML.

[16] Ledoux Hugo. Are your IndoorGML files valid? ISPRS Ann Photogramm Remote Sens Spatial Inf Sci 2020;VI-4-W1-2020:109–18. http://dx.doi.org/10.5194/isprs-annals-VI-4-W1-2020-109-2020, Conference Name: ISPRS TC IV<br>3rd BIM/GIS Integration Workshop and 15th 3D GeoInfo Conference 2020 (Volume VI-4/W1-2020) - 7&ndash;11 September 2020, London, UK Publisher: Copernicus GmbH, url: https://isprs-annals.copernicus.org/articles/VI-4-W1-2020/109/2020/.

[17] Hunter JD. Matplotlib: A 2D graphics environment. IEEE Comput Soc 2007;9:90–5. http://dx.doi.org/10.1109/MCSE.2007.55.

[18] Peng Dejian. Improving the performance of hospitals: an architectural analysis of patient journeys in China [Ph.D. thesis], Delft University of Technology; 2022, url: https://research.tudelft.nl/en/publications/improving-the-performance-of-hospitals-an-architectural-analysis-.

[19] Ruiz Francisco, Garcia Felix, Calahorra Luis, Llorente César, Gonçalves Luis, Daniel Christel, et al. Business process modeling in healthcare. Stud Health Technol Inform 2012;179:75–87.

[20] van Merode Godefridus G, Groothuis Siebren, Hasman Arie. Enterprise resource planning for hospitals. Int J Med Inf 2004;73(6):493–501. http://dx.doi.org/10.1016/j.ijmedinf.2004.02.007, url: https://www.sciencedirect.com/science/article/pii/S138650560400053X.

[21] (NWO) Dutch Research Council. NWO impact - Theory - The impact pathway, url: https://impact.nwo.nl/en/working-with-an-impact-plan/theory-the-impact-pathway.