buildings

MDPI

Article

A gbXML Reconstruction Workflow and Tool Development to Improve the Geometric Interoperability between BIM and BEM

Yikun Yang [1], Yiqun Pan [1,*], Fei Zeng [1], Ziran Lin [2] and Chenyu Li [3]

[1] School of Mechanical Engineering, Tongji University, Shanghai 201804, China; ian@tongji.edu.cn (Y.Y.); 2132745@tongji.edu.cn (F.Z.)
[2] College of Electronic and Information Engineering, Tongji University, Shanghai 201804, China; 1950291@tongji.edu.cn
[3] Tongji Architectural Design (Group) Co., Ltd., Shanghai 200092, China; 53lcy@tjad.cn
* Correspondence: yiqunpan@tongji.edu.cn

Abstract: The BIM-based building energy simulation plays an important role in sustainable design on the track of achieving the net-zero carbon building stock by 2050. However, the issues on BIM-BEM interoperability make the design process inefficient and less automatic. The insufficient semantic information may lead to results inaccurate while the error-prone geometry will terminate the simulation engine. Defective models and authoring tools lagging behind the standard often cause failures in creating a clean geometry that is acceptable to the simulation engine. This project aims to develop a workflow that helps with the documentation of a lightweight geometry in gbXML format. The implemented workflow bypasses the modeling inaccuracies and irrelevant details by reconstructing the model based on extrusions on patched floor plans. Compared with other gbXML files exported by BIM authoring tools, the resulting gbXML is more lightweight with airtight space boundaries. The gbXML has been further tested against EnergyPlus to demonstrate its capability in aiding a seamless geometry exchange between BIM and BEM.

Keywords: building information modeling; building energy modeling; gbXML; plugin development; model simplification; interoperability

Citation: Yang, Y.; Pan, Y.; Zeng, F.; Lin, Z.; Li, C. A gbXML Reconstruction Workflow and Tool Development to Improve the Geometric Interoperability between BIM and BEM. *Buildings* **2022**, *12*, 221. https://doi.org/10.3390/buildings12020221

Academic Editor: Geun Young Yun

Received: 12 December 2021
Accepted: 17 January 2022
Published: 16 February 2022

1. Introduction

Emissions from the operation of buildings reached an all-time high of 9.95 Gt CO_2 in 2019 [1], drifting off the track to a net-zero carbon building stock by 2050. The buildings were responsible for 28% (38% if adding construction [2]) of total global energy-related CO_2 emissions, which emphasizes the urgent need to reduce the energy demand, decarbonize the power sector and innovate materials. With the ever tightened sustainability goals, the building information modeling (BIM) in conjunction with the building energy modeling (BEM) provides a life-long regulation on carbon emission. BIM is a digital representation of a building entity. It serves as a shared knowledge resource for communication among architecture, engineering, construction and facility management industries [3], along with the whole life cycle of the building project. At the current developing stage, research on BIM application is revolutionized to be incorporated with other technologies, such as BEM [4]. Mutually, by tapping into the data of BIM, there is an opportunity to make BEM a time-saving, low-cost, automatic, consistent and accurate process [5]. Such BIM-based BEM permits efficient feedback among various disciplines on energy-use intensity, indoor environment and life-cycle analysis, helping with the decision-making process. With the ever-growing software ecosystem, more and more real-world practices are implementing the BIM-BEM workflow to tap into the energy-saving potential of buildings [6,7], in design, construction and operation, to meet the net-zero carbon goals.

However, the interoperability between BIM and BEM is not a seamless task, which is understood as the possibility of communication, exchange, and the use of data among

software applications [8]. Data exchange schema may act as the middle-ware between BIM authoring tools (Revit, ArchiCAD, etc.) and the building performance simulation (BPS) tools (IES-VE, EnergyPlus, etc.). Two widely acknowledged schemas are the industry foundation class (IFC) [9] and the green building extensible markup language (gbXML) [10]. IFC aims to enable information sharing and process improvement during the whole building life cycle against the context of the entire AEC industry [11]. It is comprehensive and detailed yet cumbersome in interaction. Although it could come in forms of different phases, level of details or model view definitions, there is still lots of redundant information for energy simulation, both geometric and semantic. On the contrary, gbXML is mainly developed to facilitate data transformation from BIM to BEM. Its data structure, both geometric and semantic, is in line with the requirements of simulation engine. IFC supports sweep volume, constructive solid geometry and boundary representation (B-rep) [12], while gbXML only accepts surfaces represented by boundary loops, which contributes to B-rep spaces (Figure 1a). There are two modeling conventions of gbXML: one with gaps among spaces, originated from BIM that already has the surface-matching information in store; another without gaps, which follows the manual modeling convention picking the wall center line as the separation for thermal zones (Figure 1b). Apart from the bias in the surface area, models by these two conventions remain the same in mathematics.

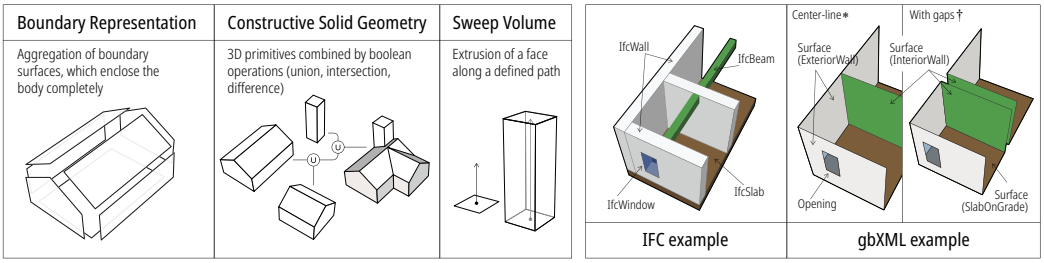

Figure 1. (**a**) Three types of geometry representation in IFC. (**b**) Different geometry representations by IFC and gbXML, an example. (∗) Modeled by wall center line (without gaps). (†) Modeled by wall faces (with gaps).

There is a semantic gap caused by insufficient simulation parameters, such as the weather data, thermal properties of components and the HVAC system. Such a gap could be bridged by energy modeling presets. Cerezo et al. issued an XML based template system, aiming to quickly set up thermal zones in the schematic or urban design explorations [13]. Moreover, the National Renewable Energy Laboratory developed the open-access building component library that is embedded in BPS tools as modeling templates. The current version of IFC and gbXML schema gradually supports the HVAC system on its topology and control logic. On the other side, the geometric gap has long been arousing discussion. The current software ecosystem still lacks robust tools to do the geometric transformation that can seamlessly connect to the simulation engine. The issue comes along with the coarsely modeled geometry not oriented for BEM, which may have components that are missing or disjoint, resulting in leaky spaces. Additionally, authoring tools are not that perfect in geometry export. Due to the information loss during data exchange and the huge number of entities in an architecture project, it is hard or even impossible to fix the geometry manually. Sometimes, the modeler has to abandon the semi-automated workflow of BIM-BEM and turn to energy modeling from scratch.

This paper mainly focuses on the geometric part and proposes an automatic workflow to reconstruct the BEM geometry to help with the BIM-BEM interoperability. Such a workflow extrudes the boundary-represented space based on corrected floor plans, which may tolerate defective model input and produce sealed spaces. To demonstrate its capability, EnergyPlus is selected as the simulation engine to test the usability of the exported geometry.

In the following discussion, the acronyms "BIM" and "BEM" are used to represent both the model itself and the modeling process. It is a common practice in the research community and by software developers that the two are used alternately [14]. The BIM authoring tool is exemplified by Revit and the simulation engine by OpenStudio/EnergyPlus, as they are well accepted by the market. The IFC used is version 4.0.2.1 in line with the norm ISO 16739-1:2018 [9] and the targeting gbXML is version 6.0.1 [10].

The rest of the paper is organized as follows: Section 2 leads a literature review, focusing on the geometry interoperability, and explains why the workflow of reconstruction is selected. Section 3 comes to the core algorithm of this approach along with pseudo code and schematic diagrams. Section 4 demonstrates the capability with model tests and remarks on its pros and cons. Section 5 is the conclusion.

2. Literature Review

Building performance simulation proves to be of great help in the design process. At the concept stage, BEM may give feedback on multiple design variants, thus achieving "performance-driven". At the development stage, a detailed load estimation by BEM may help with the design of the HVAC system, to cut down the energy use and carbon emission. BEM is also essential in standard and code compliance, such as LEED labeling. The design process calls for intensive cooperation and instant feedback. That is where the concept of BIM-based BEM comes in with the most obvious advantage: the quick and accurate generation of the energy model [15]. A BIM model has the ability to offer abundant, immutable data for energy simulation, which saves the user from gathering and recreating such information [5]. One limitation of the conventional BEM is the manual preparing process of geometry that is time consuming and highly dependent on the modeler's thermal view toward the original data source [15]. However, BIM-based BEM ensures that the automatic geometric transformation is implemented under explicit rules in the algorithm in order to prevent arbitrary data improvisation and preserve the data integrity [16]. This makes the BEM and the simulation result consistent among all users. For a seamless data exchange in BIM-based BEM, the interoperability is a crucial base.

2.1. Software Ecosystem

Over decades, the interoperability has been supported by standard data exchange schemas, such as IFC and gbXML, as well as software both academic and commercial.

As its initial purpose, the exchange schema lies at the center of BIM and BEM authoring tools. This schema-centered system demands all software interchange their data explicitly with one single BIM document. Such an architecture applies to traditional BEM authoring tools that incorporate modeling and simulation on the same platform, most of which have their own simulation engine. Some well-known examples are: Trace 700, Green Building Studio (GBS), EDSL-Tas, IES Virtual Environment (IES-VE) and OpenStudio. The geometry interoperability thereby is decided by how well the BIM is documented (by the BIM authoring tool) and accepted (by the BEM authoring tool). According to the official compatibility matrix [17], only a few tools, such as IES-VE and OpenStudio, have been validated for their support to the latest IFC format. Some tools implant algorithms for necessary geometric transformations while others may only access to the tailored geometry in gbXML. However, there is no thorough validation available for their support to gbXML. According to an official report [18], OpenStudio is the only one that has passed for now. When the transformation error either resulting from BIM export or BEM import is unknown, the modeler has to revise the geometry iteratively, then export it to BEM for checks.

To ease the iterative manual fix on geometry models, another ecosystem centered around certain computer-aided design (CAD) software has gained more popularity. The logic of geometry exchange stays the same but more implicitly, or even without IFC or gbXML. A good example is Autodesk Revit which embeds two cloud services on BPS: GBS (based on DOE-2) and Insight (based on EnergyPlus) [19]. The user may obtain an instant simulation result on model revisions in the BIM authoring interface. Under the hood,

gbXML still acts as the data schema. Another parametric toolkit, LadybugTools, features in two workflows that may access BIM information by Revit Application Programming Interface (API) other than the explicit file exchange. One is the Pollination Revit Plugin [20] that exports clean models for cloud simulation, and the other is the Revit-Rhino. Inside-Grasshopper-Honeybee tool chain that invites the BIM information into a parametric modeling environment. Such workflows could bypass the information loss during the data exchange and build up an energy model based on its own schema (hbJson [21]) more efficiently.

Other than traditional physical modeling engine, such as EnergyPlus and DOE-2, the Modelica-based BEM is a nascent field. Modelica is an equation-based, object-oriented language that is developed for component-level modeling of a physical system's topology by means of component-connection representation [22]. The LBNL Modelica library and the platform Dymola, OpenModelica, are within such an ecosystem. Andriamamonjy et al. [23] developed an automatic IFC-based workflow to build up a Modelica model for energy simulation. Kim et al. [24] implemented *Revit2Modelica* for Modelica-based BEM by developing a plugin via RevitAPI. A similar approach can be found in the research of Jeong et al. [25], which also demonstrates how portable it is to access data directly within BIM authoring tools.

In summary, the software ecosystem can be categorized by its simulation engine (EnergyPlus or OpenModelica), or by how it is organized (schema-centered or CAD-centered), or, more narrowly, by different data exchange schemas (IFC-based or gbXML-based). No matter what, the geometry interoperability persists to be an issue due to different representations in CAD and mathematics. The CAD-software-centered ecosystem is better for small projects pursuing parametric analysis while the schema-centered ecosystem is still the major choice for mega projects. More schema-centered workflows targeting EnergyPlus will be addressed in the next section, as they are representative and universal in practice.

2.2. Geometry Workflow

Figure 2 depicts the current workflows of the geometric transformation that targets the simulation engine EnergyPlus (.idf), with typical tool or practice listed in Table 1. Since OpenStudio is a graphical application built around EnergyPlus, the workflow is suited for its data format (.osm) as well. All workflows begin with BIM authoring tools that are represented by main-stream software Revit (.rvt) and ArchiCAD (.pln). The diagram can be viewed horizontally or vertically. The X-axis represents the forwarding design phases from BIM to BEM authoring, in two isolated paths according to the different documentation schema implemented: IFC or gbXML. When moving downward, there is a loss of geometry information. On quantity, it is because the BEM requires merely a subset of BIM, while on quality, as the multiple representing BIM turns into serialized coordinates of boundary loops, the richness level degrades.

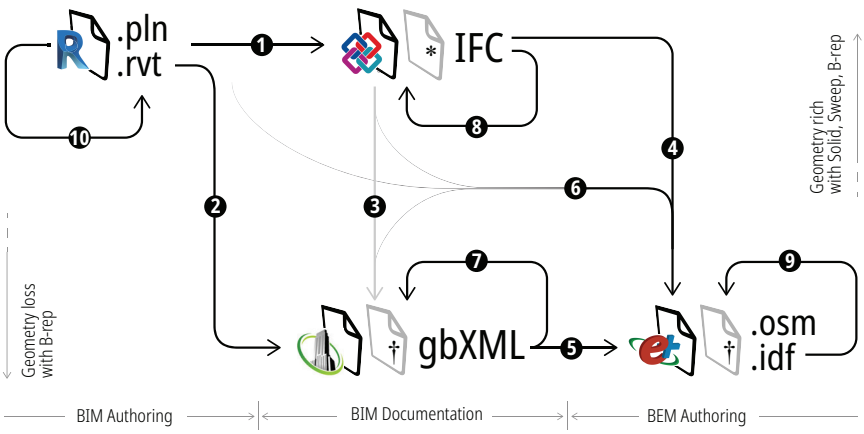

Figure 2. Typical workflows from BIM to BEM (EnergyPlus based). Notes of each indexed path in Table 1. (∗) IFC without the information of second boundary. (†) gbXML with gaps among spaces.

Table 1. Typical tool or practice implemented by the indexed workflow in Figure 2.

No.	Process	Tool/Practice	Lang.	Function	Refs.
2	gbXML authoring	Revit (Room Volumes)	C#	export gbXML by room definition	Autodesk®
		Revit (Energy Model)	C#	export gbXML by component relation	Autodesk®
3	IFC to gbXML	IFC-to-gbXML-converter	Python	with 2nd level boundary	Visschers [26]
		GST/SBT/Simergy™	C++	by calculated center-line	Bazjanac et al. [15,27–29]
4	IFC to IDF	CBIPtoIDF	-	by calculated 2nd level boundary	Giannakis et al. [30]
		OBES	-	with 2nd level boundary	Choi et al. [31]
		Green2.0	-	create center-line boundary	El-Diraby et al. [32]
5	gbXML to IDF	gbEplus	Java	implement OpenStudio reverse translator	Xu et al. [33]
		idfXML schema	-	map gbXML to IDF via idfXML	Dimitriou et al. [34]
6	BEM authoring	LadybugTools plugin	Python	modeling plugin via RevitAPI	Roudsari et al. [35]
7	IFC fixup	N/A	-	fix clashing elements, reversed surface, align space with surrounding elements	Lilis et al. [36]
		SBs generator	C#	2nd level boundary generation tolerate minor element clash convert curved wall and shadings	Ying et al. [37]
8	gbXML fixup	Spider gbXML fixer	Python	fix duplicate adjacent spaces, invalid surface type, missing gbXML attributes...	LadybugTools [38]
9	IDF fixup	EnergyPlus	C++	auto-fix sliver facet, flipped surface...	NREL
10	BIM manual fix	Solibri Model Checker	-	inspect modeling failures [27]	Solibri®

When the BIM authoring tool handles the geometric transformation and the gbXML is well documented, it is comparatively easy to implement gbXML-IDF conversion (path-5) because the two formats share the same geometry representation by vertices loop of simple polygon without holes. Many BEM authoring tools support this conversion, such as OpenStudio and DesignBuilder, and it is easy to develop, as XML and IDF are highly human-readable and explicitly defined. gbEplus, an open-sourced tool developed by Xu et al. [33], adopts the OpenStudio reverse translator to ensure a qualified geometry conversion. It also provides interface and templates for users, to inject semantic information during the translation. Dimitriou et al. [34] proposed an intermediate schema idfXML to map the objects of IDF with gbXML nodes, while acting as a template file for loads and HVAC assumptions. In a web application, Niu et al. [39] developed a Revit plugin in C# to relay the exported gbXML further into IDF. Likewise, Oh et al. [40] coded such a conversion by MATLAB. Since the workflow of gbXML shifts all calculation to the BIM side, its geometric quality determines the IDF quality.

Diverting to the workflow of IFC, if the 2nd-level boundary is accessible, it is straightforward to cast those boundaries directly as the surface of the thermal zones. This logic applies to practices such as the IFC-gbXML converter by Visschers [26], and the IFC-IDF interface by Ahn [41] and Choi [31]. It is worth noting that the 2nd-level boundaries are, by Bazjanac [42], "continuously visible regions of building element surfaces with a constant one-dimensional thermal flow rate across their area", which mandates that the boundary of the interior wall will always have its "pair" with the same shape and size. This kind of boundary is addressed as *IfcRelSpaceBoundary2ndLevel* in IFC and adopted by gbXML [10] as the space boundary.

In the case that the 2nd-level boundary is not generated or ill-documented by BIM authoring tools [28], several algorithms are available to fix it up. Rose and Bazjanac [29] presented a graph-based algorithm that traverses the pre-generated graph model that caches building elements with connection relationships to obtain the space boundary. Based on the algorithm, a tool named SBT-1 [43] is open-sourced to directly output such a boundary to a ready-to-use IDF. Giannakis et al. [30] proposed an algorithm called common boundary intersection projection (CBIP) to generate space boundaries, then deliver them to EnergyPlus or TRNSYS. The algorithm calculates the boundaries between the space object and surrounding objects, and the boundaries between building elements, then splits them into space boundaries. These algorithms require building elements correctly defined

without gaps or clashes, and space objects touching the surrounding building elements or space. Upon previous methods, Ying et al. [37] provided a more robust algorithm that could tolerate some element collisions. Additionally, it can handle curved surface, as well as shading device modeled in *ifcWall* or *ifcSlab*.

Usually, the space boundary inherited from IFC by previous methods has gaps in between. Such a geometry representation contradicts the traditional, center-line-based energy modeling convention. Nevertheless, it is allowed by simulation engines (such as EnergyPlus) given the right surface-matching relations. The net surface difference ratio between them is above 5% on average [44], which is deemed to be non-negligible in Modelica algorithms. However, in building performance simulation, center-line modeling is still the prevalent strategy with many applications. In a web-based collaborative design platform *Green2.0*, ElDiraby et al. [32] implemented several steps, including solid wall collapse and surface trim to construct center-line boundaries for OpenSudio. A similar workflow was addressed by Ladenhauf et al. [45]. They all require IFC to have full definitions on building elements and spaces without gaps or clashes.

A critical step of all workflows is the geometric transformation from the BIM elements (represented in types of Solid, Sweep or B-rep) to a B-rep space boundary in BEM. Its complexity depends on how well the BIM is documented. If rich enough, the BEM software could read out the matched-surfaces directly without the geometry manipulations that may not be their strength. In path-2, Revit has two modules for gbXML documentation. One is by "Room Volumes" that generates gbXML based on the boundary of the space and its surrounding elements. The other is by the "Energy Model" that looks for an enclosed space by building elements' relationships. If the BIM authoring tools fail to export the accurate geometry, the pressure is all on the BEM software to process all the model defects at a lower information level (path-6).

2.3. Geometry Defects

In real-world projects, gbXML is more acceptable by BEM authoring tools because it shifts the burden of geometry manipulation to the BIM side, in the documentation process. There are few reports on the BEM tool that fails to import a well-documented gbXML. Such a gbXML not only needs an official certificate of validation, but also needs to be neat without having shattered and twisted facets. When it comes to the defective gbXML, different simulation engines have different tolerance levels. In the load calculation of Trace 700, the static method may proceed given that the surfaces are clung to certain space, even though the space is not enclosed. On the contrary, EnergyPlus only accepts a space that is almost sealed. It also demands the sub-surface (corresponding to *Opening* in gbXML) to be rectangular. The gbXML must be documented by the most stringent requirements for a general application.

In the geometry transform by Revit-gbXML-OpenStudio workflow, Porsani et al. [46] reported space missing in the "Energy Model" export module of Revit. The gbXML of a small residential building has unexpected shading surfaces in place of walls or roofs. In the case of a large warehouse, the output gbXML with shattered roof terminates the simulation due to missing walls/floors. Chen et al. [47] reported a failure in representing curved walls during Revit-gbXML export. Based on actual projects and other well-documented research [28], typical failures are summarized in three levels (Figure 3), either by incapable authoring tool or unprofessional modeling.

Level-1 (Figure 3a) concerns with incapable algorithms during BIM documentation. The BIM is flawless but the geometry algorithm fails to export it correctly for BEM. There can be redundant facets generated due to gaps in the boundary surface mapping or glitches. These twisted, tiny facets may drag the radiation calculation or even sometimes terminate it by causing temperature divergence. The same situation applies with the tessellation of a curved surface. Too many shatters only burden the calculation unnecessarily. During transformation, the window has to be reshaped to a rectangular surface before it can be

accepted. Another neglect could be the equivalent shading surfaces for windows embedded in thick walls, which may affect the indoor solar gains.

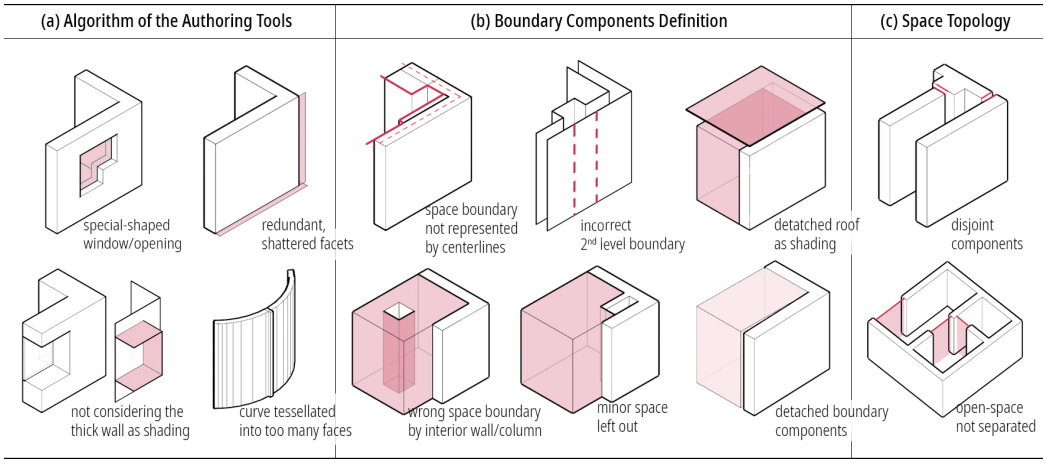

Figure 3. Typical defective BIM geometry that leads to failed gbXML export. (**a**) The export algorithm is not optimally oriented for BEM. (**b**) The space boundary components are not well defined and aligned. (**c**) The floor plan topology is incomplete.

Level-2 (Figure 3b) has ill-defined boundary components. The misalignment of space geometry and building element may cause a missing boundary or a boundary of the wrong type [28]. If the upper limit of a room is not set to the roof component, the exported gbXML will have a space whose upper face is of *Air* type and a shading surface transformed by roof.

When columns within the space act as boundary components, the exported gbXML will have holes punctuated by these columns with an exterior boundary condition. This is often the case when the integrity of the floor plan topology is violated. Another issue is about the generation of the 2nd-level boundary. When a seamless space boundary is required, it is hard for the software to extract a center line that is clean and straight, especially encountered with walls with multi-thickness or embedded columns. Under this circumstance, the 2nd-level boundary matching fails and there will be a hollow in space.

Level-3 (Figure 3c) includes the BIM that has major deficiencies in the space topology. The model lacks correct space definition. There are disjoint building elements which cannot be patched by a global tolerance. It is hard for the authoring tool to recognize enclosing spaces, not to say that the semantic information is related.

BIM-BEM interoperability is a practical topic, and one cannot expect perfect models. One reason for these flaws may originate from the documentation process of BIM. The authoring tools, either commercial or academic, are not fully targeting the requirements of BEM, and few of them have the certification on schema support. Another reason is harder to overcome: BIMs are often created without future use cases other than the generation of plans and visualizations in mind, which makes them "non-BIMs" [45].

In real-world projects, the automatic/manual fix has been a typical routine to improve the interoperability. Lilis et al. [36] provided tools that could solve inaccuracies in IFC files, which include reversed surface, clashing B-rep solid, and space object disjoint with its surrounding elements. The BEM geometry is highly serialized and hard to fix manually, even within the CAD environment. However, there can be uplifts on grammar or minor geometric flaws by scripts. The Spider gbXML fixer [38] can fix surfaces with a wrong type name, with duplicate planar coordinates and so on. The simulation engine may tolerant some errors, and EnergyPlus even corrects some automatically, such as mismatching the surface area, wrong boundary type or tilt angle. According to a testing by Fernald et al. [48], the imported gbXML gets more arbitrary diagonal surfaces after an automatic correction by IES-VE's "geometry correction" function, which in turn increases the complexity of manual revision. Such an automatic fix could be risky when the model defection is highly unpredictable. A manual fix on BIM is a way out, but sometimes it turns out to be

more tedious and time consuming than remodeling the geometry. In a trial of the Revit-gbXML-IES workflow, the exported model iterated the manual fix 10 times, reducing 10,609 unmatched edges to 121, yet was still rejected [48].

The pressing issue on BIM-BEM interoperability is to make it work with real-world projects, then make it work well. Therefore, an automatic workflow is developed to reconstruct the BEM geometry from BIM (path-2 in Figure 2), mimicking how modelers build it from scratch, to counter with the following gaps. First, the documentation of BIM lacks consideration on the simulation engine's requirements, resulting in rejection or low-efficient simulation. Second, the current geometric transformation demands BIM with space definition and joint components, which is often a luxury in real-world projects.

3. Methods

The proposed approach regenerates the floor plan by center lines of enclosing elements of BIM. Then it recognizes enclosed (or almost enclosed) polygons as space boundaries, which are further extruded vertically into B-rep spaces. Finally, by adhering openings and shadings, it serializes the geometry to standard gbXML. All geometric calculations are within the 2D plane, with float precision algorithms applied. This approach may tolerate defective models with no space definition, elements detached or overlapping and ensures that all spaces are airtight. The resulting model is more lightweight and safe for EnergyPlus without debris or voids. Figure 4 gives an overview on how geometry information is utilized and processed. Details are in three parts.

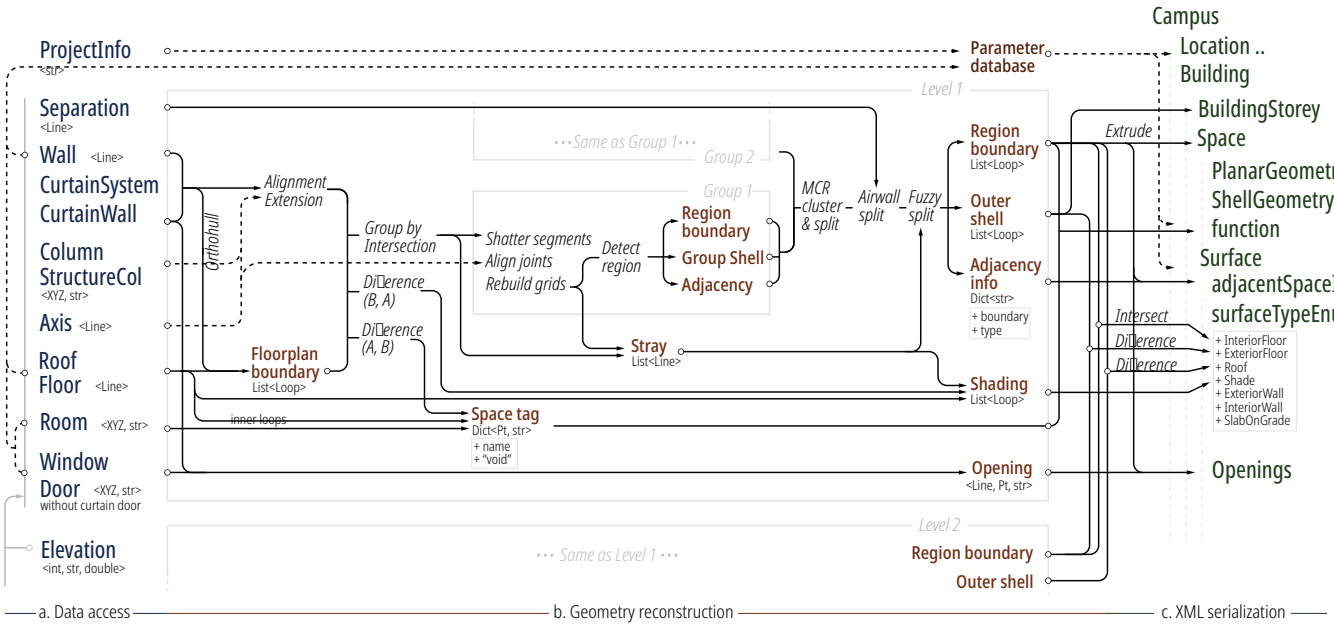

Figure 4. Overview of the workflow. (**a**) Data interface. (**b**) Geometry reconstruction. (**c**) gbXML serialization.

3.1. Data Interface

The workflow needs one semantic database and three categories of geometric information nested by each story. Some of them are listed in the left column of Figure 4. In the first category, it is a must to have a comprehensive definition of all building stories, as well as the location line and the nameplate of the enclosing structures (wall or curtain wall). If accessible, the location line of a rectangular wall is just its sweeping axis projected to the story plane (usually the center line). If not provided, it can be calculated by the intersection of the component's surface and the story planes it spans. The geometry of the floor slab is optional, including its outline and the boundary of inner holes. Since the gbXML represents geometry by vertices loop, curves are reduced to polylines by the Douglas-Peucker algorithm, which produces lesser segments than the common tessella-

tion based on curvature. The second category includes the location and nameplate of the affiliated component as well as room/space. As to a window, its label "2000 × 1400" is cached as an indexation for its geometric dimensions, such as the height, width and depth, as well as its thermal properties. The third category collects supporting lines, such as room separations, axis and grid system, to help with the alignment of geometry elements, though it is not mandatory. On top of the geometry, relative semantic information is cached within the program as a database for reference, such as the project information, thermal properties of the components and the space program settings.

A data interface is needed for the program to tap into the BIM information. As is well accepted by all software ecosystems, the IFC format has the most extensibility. There are open-sourced toolkits, such as ifcOpenShell or xbim, that can decode the IFC. However, some IFC files are ill documented because of modeling failures and the compatibility issue with BIM authoring tools. It is more ideal to retrieve the data directly from BIM authoring tools due to less information loss and more flexibility in model fixing. Software, such as Revit or ArchiCAD, all provide API to the community. The drawback is also obvious, as it is highly dependent on certain software and needs to be developed repeatedly.

3.2. Element Alignment

Floor plan patch. The reconstruction pre-processes all geometries into a hierarchical structure by Algorithm 1, floor by floor. At each *level*, all wall center lines are clustered into several *groups* by a "fuzzy" intersection check, which compares the expansion boxes of target segments (controlled by the grouping tolerance). As an example in Figure 5, the group "L1::B3::G0" will be merged with "L1::B2::G0" if the tolerance surpasses 1.84 m, because the expansion boxes are thereby intersected. Groups with their bounding rectangles overlapped are merged into a *block*. Within each block, the algorithm will try to obtain its outer boundary (marked as block shell) by Algorithm 2 based on line segments that are aligned and extended. If it fails due to highly discontinuous wall lines, the algorithm will calculate the minimum concave hull of the target set of line segments as the substitute. It starts from the orthogonal convex hull and clips inward, similar to Algorithm 3. Moreover, the floor outline outside the block shell will be severed and cast as the shading surface; the floor outline concave to the block shell will be cast as separation lines, while the remaining hollow region as "void" space, indicating an atrium, staircase, shaft or air gap. Figure 5 depicts a real-world example on this issue. The floor plan boundary consists of discontinuous lines representing the wall (in black), floor (in gray) and curtain wall (in blue/green) from inside out. Each sorted block shell has been highlighted with red dashed lines. There are 5 panes (shaded with line) clipped out as the shading surfaces, and 3 regions marked as "void" subtracted from the block shell by the outline of floor slab, representing the atrium space. The clipped floor slabs within each block are merged.

After the block shell is set, a band will be generated by offsetting it inward, controlled by the perimeter tolerance. Any endpoint of wall center lines within the band will be pulled onto the block shell (Figure 6a). This allows the curtain wall to be overwritten: any block shell segment subtracted by the "projected" wall represents the glazing area, regardless of whatsoever holds the place before. Such a process means to mend the possible space gap in the perimeter regions. The gap usually holds the space for mullions or mechanical equipments. If minor perimeter tolerance is selected, the air cavity between the curtain wall and interior wall will be modeled. However, this may lead to some unintended situations that a space "overflow" to another. In case that the wall center lines are split up by columns, there is a rule-based method to join them back together. As illustrated in Figure 6b, the center-lines will be extended and trimmed within the column outline. If failed, some axes of the outline will be added to help with the trim. They can be lines across the centroid for central symmetry shapes or center line collapsed from any polygon, but the current algorithm lacks stability when handling special-shaped columns.

Algorithm 1 Floor Boundary Patch.

Input: L_s, list of line segments L representing walls and curtain walls; P_s, list of vertices loop P representing boundaries of floor slabs;
Output: B_s, nested list of the center lines; *shades*, list of horizontal shading surface; *voids*, list of void region;

for all L in L_s **do**
  let L_{exp} be the outward expansion box for L, with the offset distance equals the grouping tolerance;
end for
cluster L in L_s by the intersection of corresponding L_{exp} into sub-list: G_s;
let G_{box} be the bounding box of G;
cluster G in G_s by the containment relations of corresponding G_{box} into parent-list: B_s;
for all B in B_s **do**
  let *shell* be the the minimum orthogonal hull of B;
  let *slabs* cache all vertices loop within *shell*;
  for all P in P_s **do**
    if P is counter clockwise **then**
      append the Boolean subtraction of P by *shell* to *shades*;
      append the Boolean intersection of P and *shell* to *slabs*;
    elseappend P to *voids*
    end if
  end for
  let $floor$ be the Boolean union of *slabs*;
  offset *shell* inward by distance equal to the perimeter tolerance, as $shell_{in}$;
  for all L in corresponding B **do**
    if endpoint Pt of L is inside *shell* and outside $shell_{in}$ **then** pull the Pt onto the *shell*;
    end if
  end for
  for all Pt in $floor$ **do**
    if Pt is inside *shell* and outside $shell_{in}$ **then** pull the Pt onto the *shell*;
    end if
  end for
  let *void* be the Boolean subtraction of *shell* by $floor$, append edges of *void* to B[0], append *void* to *voids*;
end for
return B_s, *shades*, *voids*;

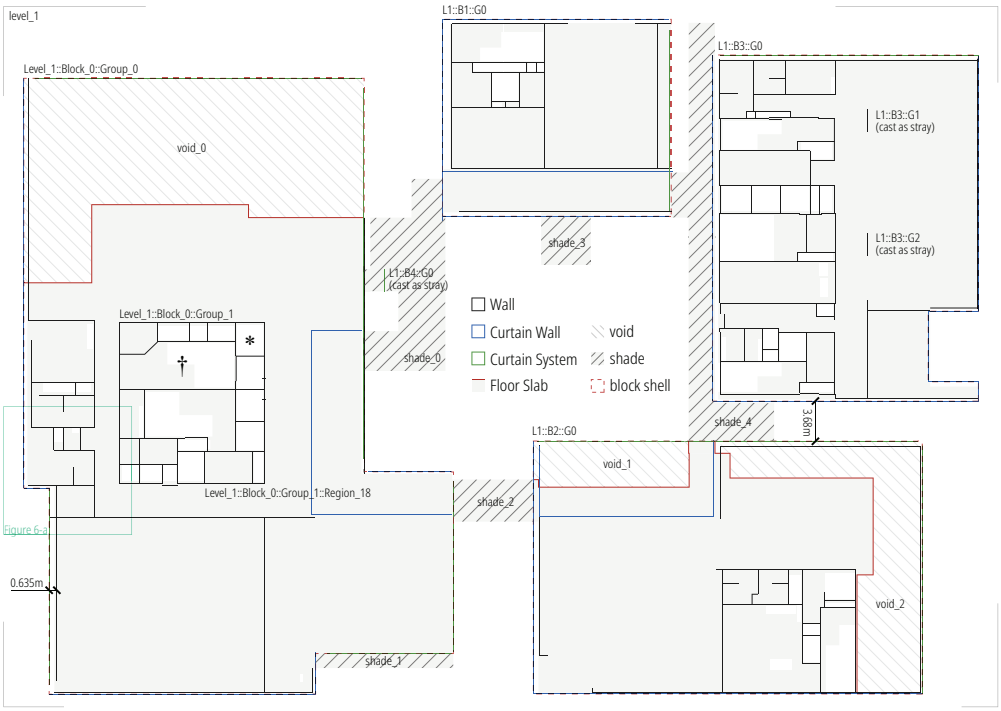

Figure 5. Different component boundaries organized into hierarchical floor plan (Level-Block-Group-Region). (∗) denotes region recognized as shaft. (†) denotes staircase.

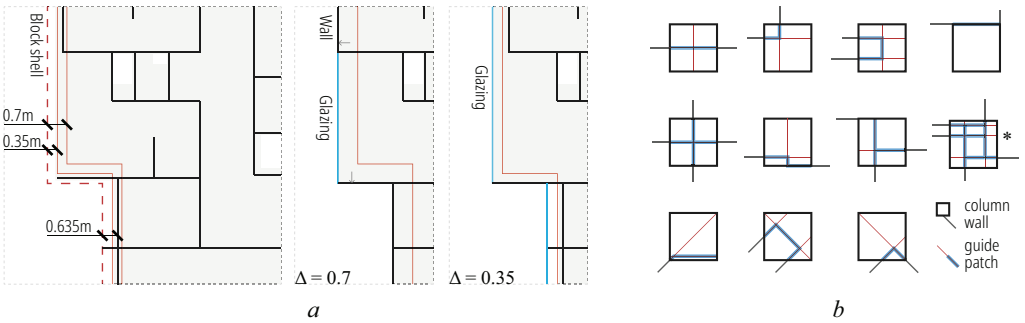

Figure 6. (**a**) Perimeter patch (Figure 5 zoomed in). Different results when pulling walls and floor boundaries to the block shell, with perimeter tolerance 0.7 and 0.35. (**b**) Column patch. (∗) denotes a situation where redundant enclosed regions are generated. They will be erased during the alignment.

Joint alignment. The defective BIM model may have walls that are overlapping, detached, misaligned or zig-zagged. Additionally, due to different wall thicknesses the extracted center lines of a continuous wall may break up naturally (Figure 7b). Joint alignment aims to fix and organize these messy line segments then feed them to the region detection. The term *joint* represents each end point with an array to store all its outward vectors by counter-clockwise angles (0~360, no duplicate), indicating its connection relations. There may be joints connected to oblique lines but often the case is the orthogonal types as listed in Figure 7c. A line segment with type-I joint is a *stray*. All strays removed will be cached for further space separation and shading extrusion.

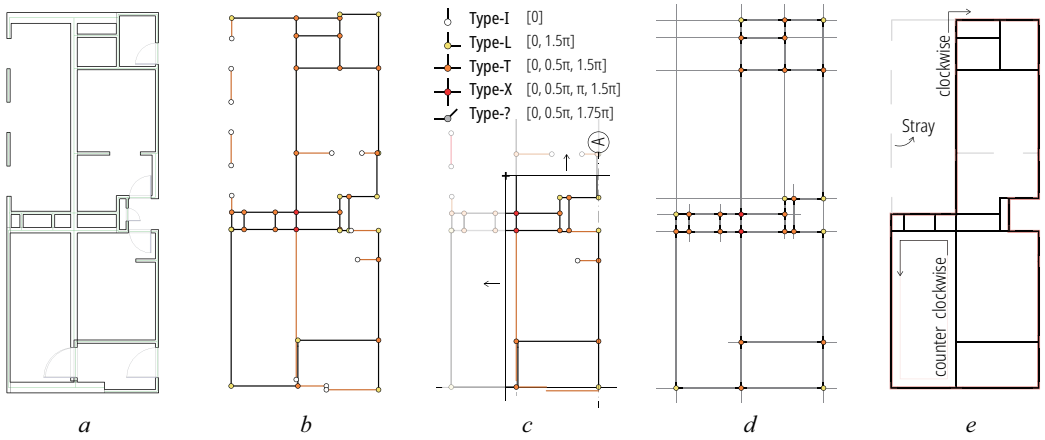

Figure 7. Process of joint alignment. (**a**) The original floor plan with wall components, viewed in Revit. (**b**) Obtain the center lines and convert the endpoints as joints. (**c**) Scan and align the joints in two orthogonal axes. The legend: typical joints and their arrays. (**d**) Rebuild the grid sorely by the joints information. (**e**) Recognize regions from the grid, resulting in clockwise shell, counter clockwise space boundary and strays.

The alignment happens within each group of (almost) intersected line segments. After breaking up all segments by their intersections, all segments are converted to joints with their corresponding arrays. There is a window with a certain width (controlled by the alignment tolerance) sliding on the X/Y-axis to locate all clusters, each of which frames out as many joints as possible. If there is an axis that happens to lie within the range of a cluster, all joints within the cluster are projected onto that axis. If not, a new axis will be generated based on the most frequently occurring joint position (Figure 7c). The axis is either from the global grid system of the document, or from the newly generated ones of the previous floor plan. The alignment applies the following operation rules on the array of joints: (1) the joint array has no duplicate vector; (2) merge the array of coincident joints; (3) erase the vector in line with the moving direction of joint A. If A is received by joint B, then erase the reverse vector within B additionally (the segment collapse situation); (4) when joint A is

projected to axis without overlapping other joints, add outward vectors to A according to its position relative to other joints projected to the axis, creating new connections; and (5) remove joints with only two opposite vectors (the redundant point situation). Note that the current alignment only works with the orthogonal grid system and allows for a global rotation by a certain angle.

A grid of lines is rebuilt based on the aligned joints. For each outward vector of joint A, the algorithm will search for joint B that has the opposite vector as well as the shortest distance to A on the direction of that vector. If joint B pairs A in the same way, a line segment is created between A and B. At the final step, all strays are removed recursively until there is no type-I joint left. The resulting grid is shown in Figure 7d.

3.3. Region Detection

After being doubled with their reversed copy, the recreated grid lines are actually composed by counter-clockwise boundary loops of each *region*, and one clockwise outer *shell* enclosing all of them. To obtain the boundary loop, the algorithm starts from any joint A and picks one outward segment, then walks until the next joint B and chooses the next outward segment of B that has the minimum counter-clockwise angle with the previous segment. The walked paths are consolidated to one closed region if the iteration comes back to the starting joint A. All space boundaries emerge when all joints are traversed (Algorithm 2). Among the results, the clockwise loop is the outer shell for this group of regions (Figure 7e). In this process, there should be unique global identifiers added to each region and its boundary edges. Since the duplicate segments are generated in pairs, it is easy to retrieve the paired edge and the identifier as its adjacency information, which helps with the surface matching in the XML serialization.

Algorithm 2 Region detection.

Input: The list of line segments, *segs*;
Output: The nested list of each region vertices loop, *regions*;
repopulate *segs* by splitting all line segments by their intersections
repopulate *segs* by adding the reversed duplicate next to each line segment
initialize List<Point> *joints* representing all unique joint point of segments in segs
initialize Dict<Point, List<Line» *nextSegs* caching all outward segments id for each joint
for all *seg* in *segs* **do**
    let *startPt* be the start point of *seg*;
    if *joints*.Contains(*startPt*) **then** *nextSegs*[*startPt*].Add(*seg*);
    else*joints*.Add(*startPt*)
    end if
end for
for all key-value pair *kvp* in *nestSegs* **do**
    if (*kvp*.Value.Count == 1) **then** remove *kvp* and its reversed duplicate;
    end if
end for
while *segs*.Count > 0 **do**
    let *thisSeg* be any segment in *segs*, let *startPt* be its start point;
    initialize List<Line> *region* in *regions* to cache all boundary edges;
    while *thisSeg*.EndPoint $\neq$ *startPt* **do**
        for all *seg* in *nextSegs*[*thisSeg*.EndPoint] **do**
            let *nextSeg* be the one with the smallest counter-clockwise angle with *thisSeg*;
        end for
        add *thisSeg* to *region*; remove it from *segs* and *nextSegs*;
        thisSeg = *nextSeg*;
    end while
end while
for all *region* in *regions* **do**
    sort segments direction in *region*;
    if *region* is closed and counter-clockwise **then** mark it as space;
    end if
    if *region* is closed and clockwise **then** mark it as outer shell;
    end if
    if *region* is open **then** mark it as stray;
    end if
end for
return *regions*;

Among all regions within a block, the algorithm will check their containment relationships and form a hierarchical data tree (if possible). Outer shell of a group and its enclosing region loop contribute to a multiply connected region (MCR) that has holes in

it. With all regions of the floor plan generated, the algorithm checks if they contain room separation line or stray line that could inform further space separations. In case that a region encloses a quite large hole in the merged floor slab, the boundary of such a hole will turn into separation lines and the region of this hole into a "void" region. Qualified line segments and the region boundary will perform the extension and polygon split, the resulting interface of the adjacent space will be assigned with the *Air* type in gbXML.

Tessellation on MCR or non-convex regions. The remaining MCRs (if exist) must be divided into simply connected regions before it can be represented by loops of vertices, which is the accepted data format of gbXML and IDF. In order to make the model as lightweight as possible, the algorithm only divides the MCR surface of the original space other than separating it into sub-zones. In this case, the IDF must inherit the area/volume value in the gbXML data to prevent an exaggerated internal load. Given an orthogonal polygon, the algorithm prefers rectangular tessellation in an architectural context. Its simplest form is to take the shortest edge as the starting point, then keep clipping the original polygon by the bounding rectangle extruded from this edge, until there is nothing left (Algorithm 3). Figure 8 depicts the process schematically. EnergyPlus also mandates that the casting surface in shadowing calculation, and the receiving surface applied with the solar distribution calculation method "Full Interior" must be convex. The workflow ensures the convexity of all casting surfaces (such as a shading device) by rectangular tessellation given orthogonal polygons, and will switch to the Hertel-Mehlhorn algorithm to solve more general cases by triangulation. The convex subdivision on interior floor surface will not be performed by default, since EnergyPlus 9.4 brings about a new method called "Pixel Counting" that could handle concave polygons.

As a quick review of notions, there are four hierarchical levels of line segments. *Level* clusters lines by their floor plan and extrusion height. *Block* discriminates the inside and outside of a building. *Group* represents a cluster of adjoining regions that are simply connected, and a *region* stands for the 2D outline of a space. There are three tolerances controlling the reconstruction process. The grouping tolerance affects the recognition of building blocks. If tuned too large, several isolated podium blocks will be merged into one. The perimeter tolerance (usually 0~1 m) controls the erasing of sliver space at the perimeter area. The alignment tolerance affects the accuracy of floor plan recognition. Usually the value is half the average wall thickness (0.1~0.2 m). These tolerances are within a narrow range due to human scale in architecture design, but they may overflow due to modeling errors.

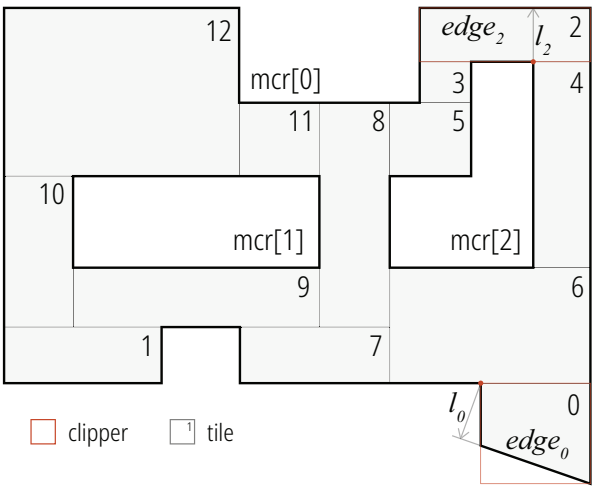

Figure 8. Rectangular Tessellation.

Algorithm 3 RectangularTessellate(P)

Input: The list of vertices loop representing MCR: *P*;
Output: The list of vertices loop representing each rectangular tile: *Ts*
let the clockwise loop in *P* be *shell*;
let the counter-clockwise loops in *P* be *holes*;
initialize vertices loop *clipper*
while P.Count == 1 and P[0].Count <= 5 **do**
  for all *thisPt* in *shell* **do**
    let *thisEdge* be the line segment from *thisPt* to the next point;
    let θ_1, θ_2 be the inward angle of endpoints of *thisEdge*;
    if $\theta_1 < 180$ and $\theta_2 < 180$ and *thisEdge* has the minimum length **then**;
      let *cutPt* be the endpoint of adjacent edges of *thisEdge* that has the minimum distance to *thisEdge*;
      let *clipper* be the bounding rectangle of *thisEdge* and *cutPt*;
    end if
  end for
  for all *hole* in *holes* **do**
    if *hole* inside or intersected with *clipper* **then**
      locate *cutPt* in *hole* that has the minimum distance to *thisEdge*;
      let *clipper* be the bounding rectangle of *thisEdge* and *cutPt*;
      add boolean intersection of *clipper* and *shell* to *Ts*;
    end if
  end for
  let *P* be the boolean subtraction of *P* by *clipper*;
  do RectangularTessellate(P) recursively;
end while

3.4. Geometry Regeneration

The spaces will be extruded vertically based on the region boundaries, level by level, while creating the belonging walls and their adjacency relationships. The walls are only discriminated by their type (*ExteriorWall*, *InteriorWall* or *Air*). The walls extruded by separation lines are deemed to be airwalls. Thermal properties are unified among interior walls due to their limited implication on the simulation result. The space, by default, is named after its belonging group and level. If provided with a room identification label (such as "office-1" or "void") and its location point, the program will allocate the nameplate to the space enclosing that point. Additionally, he algorithm offers limited function guessing for anonymous spaces. For example, crooked corridors tend to have larger normalized perimeter-area ratio, which isolates them from the regular-shaped office rooms. By calculating the area ratio of a space and the hole of the floor it encloses, the spaces can be roughly discriminated as shaft or staircase($*$, † in Figure 5). Specifically, the space assigned with "void" label, as well as the shaft and staircase, will have its floor surface modeled with *Air* type. It can be further merged with its adjacent space at lower level if stack effect is considered in the simulation.

The creation of floor surfaces contains the following three steps: (1) create the top-level roof and the ground floor by the region loops in the corresponding floor plan; (2) create the roof and exposed floor on middle levels by subtracting the region loops by the outer block shells of the adjacent floor plan; and (3) create the rest of the floor surfaces by iterating the Boolean intersection of all regions between two adjacent floor plans. The surfaces are named in pairs during Boolean operation to ensure that they are matched.

Windows and doors are attached to surfaces as openings by retrieving the wall line segment that is closest to the location point (or location line in terms of curtain wall). In reconstructing the glazing, all mullions are ignored. The opening vertices loop is manipulated within its hosting surface because EnergyPlus only accepts rectangular sub-surfaces as openings (Figure 9). (a) For special-shaped, orthogonal polygon, tessellate it with the aforementioned Algorithm 3. (b) For a non-orthogonal special-shaped polygon, redraw its bounding rectangle with an equivalent area by adjusting the height. (c) For collided components, obtain the rectangular tessellation of their Boolean union region. (d) For windows spanning multiple levels, only pick its Boolean intersection with the current hosting surface. By doing this, the algorithm avoids failures in BEM authoring that are caused by windows/doors with an irregular shape. The opening or hole in the wall is ignored.

Although the gbXML schema allows the shading surface to be clung to a certain space, the workflow here will append the shading surface directly to the site regardless of

interior ones. Three kinds of components will be transformed to surface with "Shade" type: any specifically defined shading device, the floor slab and wall out of the extruded space. After the region detection, stray line segments outside the block shells will be extruded as vertical shadings. The floor slab intersected with extruded spaces will be clipped by the corresponding block shells as horizontal shadings (the slab may be on the floor level or floating). In addition, there could be degenerated space whose upper surface is neither fully covered by the space nor the floor/roof slab above. Such a space will be removed leaving its exterior wall cast as extra shadings.

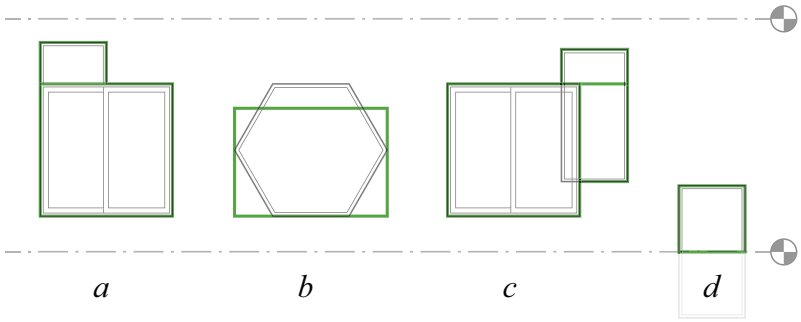

Figure 9. Opening simplification. (**a**) Orthogonal polygon. (**b**) Special-shaped. (**c**) Overlapping. (**d**) Spanning multiple levels.

XML serialization. The algorithm has already processed all geometric information needed for serialization, and cached them within the internal classes that correspond to each element of gbXML schema, as illustrated in Figure 3. As for the semantic part, project information, space program settings, thermal property of various components extract from the data interface will be packed as the simulation presets attached to gbXML for reference. The serialization is performed by the document object model, following the latest gbXML schema 6.0.1. The output has passed the official level-1 standardization test [49].

4. Results and Discussion

To demonstrate the effectiveness of the modeling approach, software is developed in C# to implement all reconstruction and serialization process, targeting gbXML 6.0.1. The software depends on an open-source package *Clipper* [50] to ensure the stability in Boolean operations. For the ease of testing, a plugin *Gingerbread* is developed as a data interface to BIM authoring tools, which is an open-source project on GitHub [51]. Its first release suits for Revit 2020. The native gbXML export modules, "Room Volumes" and "Energy Model" of Revit are the best alternatives for comparison. Figure 10 shows the user interface of this plugin. The main dialog allows the user to customize the header information and sets up modeling tolerances. The tolerance controls the resolution how floor plans are recognized and aligned. Additional options can be switched on and off with the following check boxes. "Include linked Revit model" allows to retrieve component information from linked external models within current environment. "Shadow previous floor plan" indicates that the floor plan will guide the alignment of next floor to reduce sliver facets during surface matching. The embedded Aragog-gbXML-Viewer-R14 [52] offers visualized feedback to the user so that they can tune the tolerance or fix the model immediately.

The tool is tested against two real-world BIM projects. For replication convenience, case A is a technical school, an official BIM example shipped with Revit (Figure 10). Case B is an academic building with several blocks of podiums, which are roughly modeled based on CAD drawings without any room definition. As Revit 2020 does not support the modeling of shading elements, they are modeled by floor slabs or walls instead. The floor plan topology of the two projects are rather complex, including large and small offices, shafts, stairs and atrium voids. Components are varied, such as curtain system, slope roof, skylights and parapets. All project models come in three versions of gbXML that are exported respectively by this tool, the "Room Volumes" module, and the "Energy Model"

module of Revit. Each gbXML is further processed by OpenStudio-3.2 to IDF with default simulation settings, then tested against the EnergyPlus-9.5 to check if the geometry is acceptable. The sample files and the test results are cached in the repository of this project. In the palette of Aragog-gbXML-Viewer-R14, dark red represents the roof, and pale yellow denotes the shading surface. Interior floors are cyan, and the interior walls are green. The glazing will overlay a translucent gray on the graphics behind.

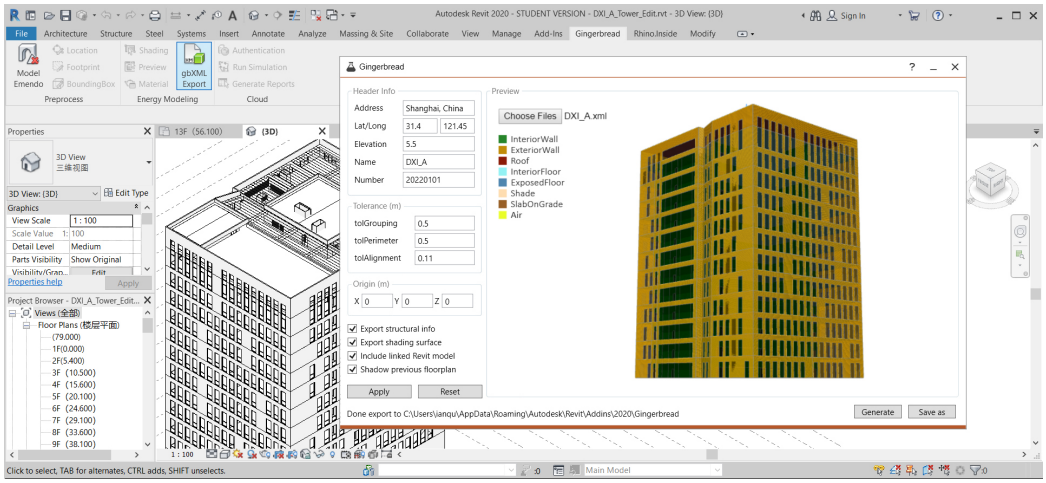

Figure 10. Interface of the plugin.

Figure 11 depicts the results of case A. Model A-a exported by this tool stands out for its sealed and neat geometry, while model A-b confuses the type of some surfaces, by taking the roof as shading and ceiling on the first level as the roof. When it comes to the ill-defined plenum space, Revit regards it as a void gap with an adiabatic boundary. In addition, the shading surfaces (modeled as floor slab) are not translated. Model A-c just has too many shattered surfaces, such as the crinkly roof or the zig-zagged floor boundary, usually resulting in spaces that are not enclosed. Additionally, the geometry is much too detailed in the description of mullions (as shading surface), which hardly contributes to the simulation accuracy but only slows down the radiation calculation. The geometry of model A-a is the only one that has been accepted by OpenStudio without errors, compared with the other two. By focusing on the integrity of space boundary, it eliminates unnecessary facets and compresses the file size from 29.1 M to 3.14 M, with only 492 surfaces compared to the original 1400 ones.

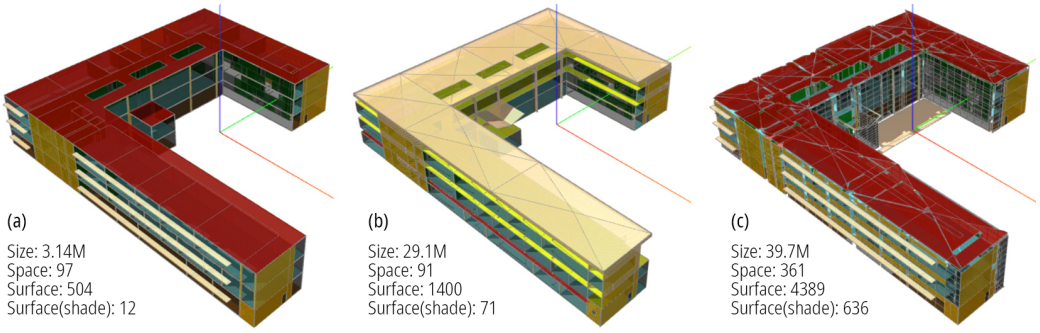

Figure 11. Exported gbXML by different modules (case A) (**a**) By this tool. (**b**) By "Room Volumes" in Revit. (**c**) By "Energy Model" in Revit.

Likewise, three gbXML models are generated for case B, as shown in Figure 12. In order to obtain the model B-b exported by the "Room Volumes", the user has to pre-process the BIM, in the authoring tool, adding room manually or by scripts. This is often the case that the BIM in actual projects lacks a comprehensive definition of the floor plan

topology. Case B has decorative curtain systems outside the exterior wall. Model B-b gets bewildered dealing with the gap between them, and fails in expressing the glazing panels. Similar to case A, there are surfaces assigned with the wrong type (exterior wall as shading/adiabatic surface). As for the topology integrity, it is highly dependent on how rooms/spaces are defined by the modeler. Model B-c is generated by the "Energy Model" module that can detect space automatically. However, it fails to recognize enclosed (or almost enclosed) spaces as many as possible, especially in the podium, which cripples the floor plan topology. The surfaces not assigned to certain space are all turned into shading surfaces. Unlike the situation of case A, model B-c translates no curtain system due to the failure in recognizing it as the space boundary. The gbXML documentation process of BIM authoring tools performs detailed 3D boundary checking and maps every surface to XML, which lacks robustness when dealing with defective models, thus making it hard for geometry exchange in real-world BIM-BEM projects. By ensuring an error-less geometry, model B-a performs better when dealing with defective BIM. It represents more spaces with less surfaces, contributing to an intact floor plan without any holes, while the other two are rejected by EnergyPlus for their leaky spaces and facet debris.

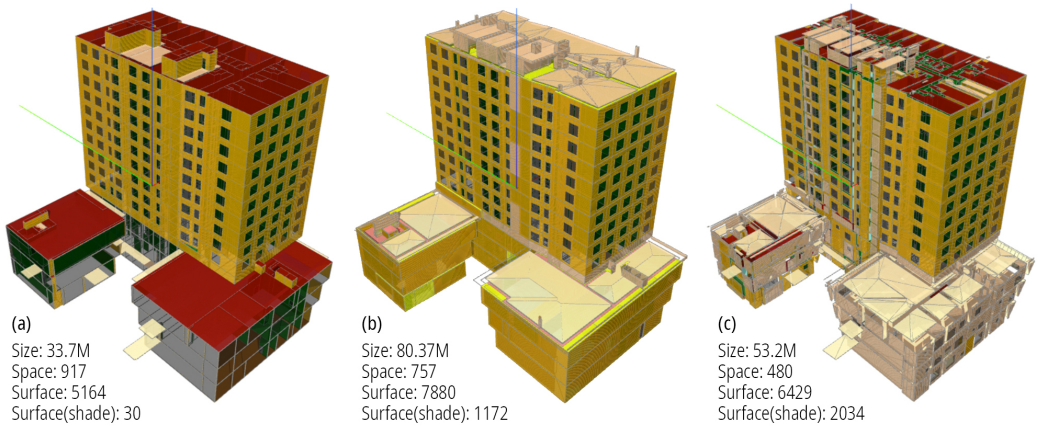

Figure 12. Exported gbXML by different modules (case B). (**a**) By this tool. (**b**) By "Room Volumes" in Revit. (**c**) By "Energy Model" in Revit.

However, there are some limits to the proposed workflow: (1) By the modeling logic of vertical extrusion, it cannot handle an oblique wall or pitched roof, as well as projects with a split-level or sloping slab, because the extrusion distance equals a certain space between the two recognized floors. This logic hinders the tool from passing the official gbXML level-2 certification, in specific testing cases such as multiple ceiling levels, vaulted ceiling with dormer and the atrium space (Figure 13). However, it is suited for common office/apartment projects. The low computational cost and smaller size will make it a useful alternative, bridging BIM and BEM. (2) The algorithm applies the same construction type to all interior walls or exterior walls. The detailed information as to the thickness and thermal properties are unified during the reconstruction of the floor plan. As a common practice, different interior wall types will not be discriminated by most of the BEM authoring tools, for their limited impact to the whole building's simulation result. However, the gbXML will not describe any columns embedded in the exterior walls, which can be a drawback when thermal bridging is considered. (3) The workflow aims to tolerate as many modeling errors as possible and to ensure an error free documented BIM (in gbXML) that can be accepted by simulation engine. Therefore, it is obscure about the boundary on what kind of model can be processed and what cannot. Especially when facing real-world modeling failures, it needs more tests to clarify detailed requirements on inputs. Currently, the effectiveness still depends on how good the BIM is (the "garbage in, garbage out" principle), and there is no guarantee on a perfect output every time.

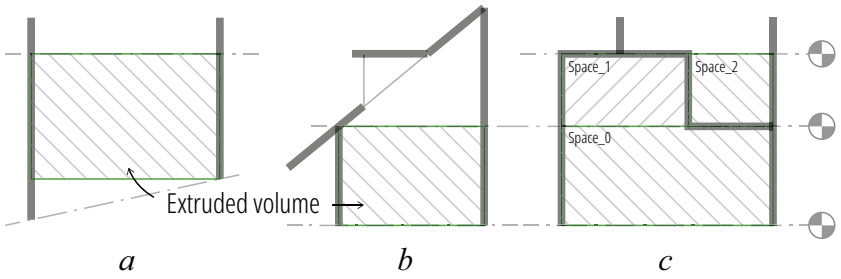

Figure 13. Failed extrusion. (**a**) With sloping slab on grade. (**b**) With pitched roof and dormer/skylight. (**c**) Split-level house.

5. Conclusions

In this paper, an automatic workflow is proposed to reconstruct the building geometric model based on the extrusion of room boundaries. Such a workflow is of less computational cost and more robust to defective BIMs. The resulting geometry is boundary represented and airtight, with no gaps among spaces. On top of that, it has fewer surface elements while keeping the integrity of the original model, which may increase the speed and stability of the energy simulation. When tested against two real-world projects, this workflow demonstrates its capability in exporting lightweight and clean gbXML models that can be accepted by EnergyPlus. Other gbXML files authored by Revit fail. Although the workflow has some limitations, it could serve as an alternative to the gbXML crafting, when facing medium or large projects that are not documented very well.

The factor of the modeler should not be neglected in the interoperability of BIM-BEM. Although much work has been done to bridge the gap, with software cooperating seamlessly, standard and schema up to date, it is still a struggle to perform BIM-based energy simulation. It probably relates to a specific time of the evolving BIM market, when most of the models are not established during the forward design process but transformed from CAD drawings. The quality of BIM cannot be guaranteed when it is moved away from the decision table and acting merely as a visualization tool. There needs to be more standards and guidelines to regulate the modeling process so that the BIM-BEM interoperability can be improved on a solid basis, as well as a more robust workflow and software for support.

The gbXML reconstruction approach proposed in this paper is a practical attempt. There are needs for preliminary energy simulation in real-world projects, such as concept comparison in parametric design and performance evaluation for code compliance. The developed tool may help with its applications on defective BIMs and potentially save human labor. The rebuilding workflow follows the method how professionals build up models from scratch. That is where artificial intelligence could come in and bring about more intuitive solutions on fixing defective BIMs. The neural network has demonstrated its ability in pixel-based floor plan recognition, and its application in the vector world is one intriguing topic.

The author will continue the development of the tool, aiming to increase its robustness on defective yet regular-shaped models and improve the user interface. In case of model failures beyond the capability of the algorithm, a graphical interface is needed to give instant feedback on potential improvements of the floor plan. To be more robust, a smarter algorithm on fuzzy space recognition is needed. In addition to that, an extrusion algorithm with adaptive shape that suits the boundary will be of great help in handling pitched roof and split-level structures.

Author Contributions: Conceptualization, Y.Y., Y.P. and C.L.; Methodology, Y.Y. and F.Z.; Project administration, Y.P. and C.L.; Software, Y.Y., F.Z. and Z.L.; Supervision, Y.P. and C.L.; Writing—original draft, Y.Y.; Writing—review & editing, Y.P. and Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China (grant number 51978481).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
BIM	Building Information Modeling/Model
BEM	Building Energy Modeling/Model
BPS	Building Performance Simulation
B-rep	Boundary Representation
CAD	Computer-Aided Design
gbXML	Green Building Extensible Markup Language
HVAC	Heating, Ventilation and Air-Conditioning
IDF	Input Data File (of EnergyPlus)
IFC	Industry Foundation Class
MCR	Multiply Connected Region

References

1. Tracking Buildings 2020. Available online: https://www.iea.org/reports (accessed on 11 December 2021).
2. UN Environment Programme. *The 2020 Global Status Report for Buildings and Construction: Towards a Zero-Emissions, Efficient and Resilient Buildings and Construction Sector*; Technical Report for United Nations Environment Programme: Nairobi, Kenya, 2020.
3. Eastman, C.M. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*; Wiley: Hoboken NJ, USA, 2008; pp. 21–23.
4. Liu, Z.; Lu, Y.; Peh, L.C. A Review and Scientometric Analysis of Global Building Information Modeling (BIM) Research in the Architecture, Engineering and Construction (AEC) Industry. *Buildings* **2019**, *9*, 210. [CrossRef]
5. Gao, H.; Koch, C.; Wu, Y. Building information modelling based building energy modelling: A review. *Appl. Energy* **2019**, *238*, 320–343. [CrossRef]
6. Carvalho, J.; Almeida, M.; Bragança, L.; Mateus, R. BIM-Based Energy Analysis and Sustainability Assessment—Application to Portuguese Buildings. *Buildings* **2021**, *11*, 246. [CrossRef]
7. González, J.; Soares, C.A.P.; Najjar, M.; Haddad, A.N. BIM and BEM Methodologies Integration in Energy-Efficient Buildings Using Experimental Design. *Buildings* **2021**, *11*, 491. [CrossRef]
8. Abanda, F.H.; Byers, L. An investigation of the impact of building orientation on energy consumption in a domestic building using emerging BIM (Building Information Modelling). *Energy* **2016**, *97*, 517–527. [CrossRef]
9. ISO 16739-1:2018. Available online: https://www.iso.org/standard/70303.html (accessed on 11 December 2021).
10. gbXML 6.0.1 Schema. 2020. Available online: https://www.gbxml.org/schema/6-01/GreenBuildingXML_Ver6.01.xsd (accessed on 11 December 2021).
11. Dong, B.; Lam, K.P.; Huang, Y.C.; Dobbs, G.M. A comparative study of the IFC and gbXML informational infrastructures for data exchange in computational design support environments. In Proceedings of the 10th Conference of International Building Performance Simulation Association, Beijing, China, 3–6 September 2007.
12. Donkers, S. Automatic Generation of CityGML LoD3 Building Models from IFC Models. Master's Thesis, Delft University of Technology, Delft, The Netherlands, 2013.
13. Reinhart, C.F.; Cerezo Davila, C. Urban building energy modeling—A review of a nascent field. *Build. Environ.* **2016**, *97*, 196–202. [CrossRef]
14. Turk, Ž. Ten questions concerning building information modelling. *Build. Environ.* **2016**, *107*, 274–284. [CrossRef]
15. Bazjanac, V. IFC BIM-Based methodology for semi-automated building energy performance simulation. In Proceedings of the 25th International Conference on Information Technology in Construction, Santiago, Chile, 15–17 July 2008.
16. Bazjanac, V. Implementation of semi-automated energy performance simulation: Building geometry. In *Managing IT in Construction/Managing Construction for Tomorrow*; CRC Press: Boca Raton, FL, USA, 2009; pp. 613–620.
17. Certified Software. Available online: https://www.buildingsmart.org/compliance/software-certification/certified-software (accessed on 11 December 2021).

18. Software Tools That Integrate with gbXML. Available online: https://gbxml.org/Software_Tools_that_Support_GreenBuildingXML_gbXML (accessed on 11 December 2021).
19. Insight: Building Performance Analysis Software. Available online: https://www.autodesk.com/products/insight/overview (accessed on 11 December 2021).
20. Pollination Revit Plugin. Available online: https://pollination.cloud/revit-plugin (accessed on 11 December 2021).
21. Honeybee Schema. Available online: https://www.ladybug.tools/honeybee-schema (accessed on 11 December 2021).
22. Fritzson, P.A. *Principles of Object Oriented Modeling and Simulation with Modelica 2.1: A Cyber-Physical Approach*; Wiley: Hoboken, NJ, USA, 2015; pp. 19–20.
23. Andriamamonjy, A.; Saelens, D.; Klein, R. An automated IFC-based workflow for building energy performance simulation with Modelica. *Autom. Constr.* **2018**, *91*, 166–181. [CrossRef]
24. Kim, J.B.; Jeong,W.; Clayton, M.J.; Haberl, J.S.; Yan,W. Developing a physical BIM library for building thermal energy simulation. *Autom. Constr.* **2015**, *50*, 16–28. [CrossRef]
25. Jeong, W.; Kim, J.B.; Clayton, M.J.; Haberl, J.S.; Yan, W. Translating building information modeling to building energy modeling using model view definition. *Sci. World J.* **2014**, *2014*, 638276. [CrossRef] [PubMed]
26. Visschers, M. BIM Based Whole-Building Energy Analysis towards an Improved Interoperability: A Conversion from the IFC File Format to a Validated gbXML File Format. Master's Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2016.
27. O'Donnell, J.; Maile, T.; Rose, C.; Mrazovic, N.; Morrissey, E.; Regnier, C.; Parrish, K.; Bazjanac, V. *Transforming BIM to BEM Generation of Building Geometry for the NASA Ames Sustainability Base BIM*; Technical Report for Lawrence Berkeley National Laboratory: Berkeley, CA, USA, 2013.
28. Maile, T.; O'Donnell, J.; Bazjanac, V.; Rose, C. BIM Geometry modelling guidelines for building energy performance simulation. In Proceedings of the 13th Conference of International Building Performance Simulation Association, Chambéry, France, 26–28 August 2013.
29. Rose, C.; Bazjanac, V. An algorithm to generate space boundaries for building energy simulation. *Eng. Comput.* **2015**, *31*, 271–280. [CrossRef]
30. Giannakis, G.; Lilis, G.; Carcia, M.; Kontes, G.; Valmaseda, C.; Rovas, D. A methodology to automatically generate geometry inputs for energy performance simulation from IFC BIM models. In Proceedings of the 14th Conference of International Building Performance Simulation Association, Hyderabad, India, 7–9 December 2015.
31. Choi, J.; Shin, J.; Kim, M.; Kim, I. Development of openBIM-based energy analysis software to improve the interoperability of energy performance assessment. *Autom. Constr.* **2016**, *72*, 52–64. [CrossRef]
32. El-Diraby, T.; Krijnen, T.; Papagelis, M. BIM-based collaborative design and socio-technical analytics of green buildings. *Autom. Constr.* **2017**, *82*, 59–74. [CrossRef]
33. Xu, W.; Chong, A.; Lam, K.P.; Wang, H. A New BIM To BEM Framework: The Development And Verification Of An Open-source gbXML To EnergyPlus Translator For Supporting Building Life Cycle Performance Analysis. In Proceedings of the 16th Conference of International Building Performance Simulation Association, Rome, Italy, 2–4 September 2019.
34. Dimitriou, V.; Firth, S.K.; Hassan, T.M.; Fouchal, F. BIM enabled building energy modelling development and verification of a GBXML to IDF conversion method. In Proceedings of the 3rd IBPSA-England Conference BSO 2016, Great North Museum, Newcastle, 12–14 September 2016.
35. Roudsari, M.S.; Pak, M. Ladybug: A parametric environmental plugin for grasshopper to help designers create an environmentally-conscious design. In Proceedings of the 13th Conference of International Building Performance Simulation Association, Chambéry, France, 26–28 August 2013.
36. Lilis, G.; Giannakis, G.; Rovas, D. Detection and semi-automatic correction of geometric inaccuracies in IFC files. In Proceedings of the 14th Conference of International Building Performance Simulation Association, Hyderabad, India, 7–9 December 2015.
37. Ying, H.; Lee, S. Generating second-level space boundaries from large-scale IFC-compliant building information models using multiple geometry representations. *Autom. Constr.* **2021**, *126*, 103659. [CrossRef]
38. Spider gbXML Fixer. Available online: https://github.com/ladybug-tools/spider-gbxml-fixer (accessed on 11 December 2021).
39. Niu, S.; Pan, W.; Zhao, Y. A BIM-GIS Integrated Web-based Visualization System for Low Energy Building Design. In Proceedings of the 9th International Symposium on Heating, Ventilation and Air Conditioning (ISHVAC) and the 3rd International Conference on Building Energy and Environment (COBEE), Tianjin, China, 12–15 July 2015.
40. Oh, S.; Kim, Y.; Park, C.; Kim, I. Process-driven BIM-based optimal design using integration of EnergyPlus, generic algorithm, and pareto optimality. In Proceedings of the 12th Conference of International Building Performance Simulation Association, Sydney, Australia, 14–16 November 2011.
41. Ahn, K.U.; Kim, Y.J.; Park, C.S.; Kim, I.; Lee, K. BIM interface for full vs. semi-automated building energy simulation. *Energy Build.* **2014**, *68*, 671–678. [CrossRef]
42. Bazjanac, V. Space boundary requirements for modeling of building geometry for energy and other performance simulation. In Proceedings of the CIB W78, Cairo, Egypt, 16–19 November 2010.
43. EETD-SBT. Available online: https://bitbucket.org/berkeleylab/eetd-sbt (accessed on 11 December 2021).
44. Bazjanac, V.; Maile, T.; Nytsch-Geusen, C. Generation of building geometry for energy performance simulation using Modelica. In Proceedings of the BauSIM2016, Dresden, Germany, 14–16 September 2016.

45. Ladenhauf, D.; Battisti, K.; Berndt, R.; Eggeling, E.; Fellner, D.W.; Gratzl-Michlmair, M.; Ullrich, T. Computational geometry in the context of building information modeling. *Energy Build.* **2016**, *115*, 78–84. [CrossRef]
46. Bastos Porsani, G.; Del Valle de Lersundi, K.; Sánchez-Ostiz Gutiérrez, A.; Fernández Bandera, C. Interoperability between Building Information Modelling (BIM) and Building Energy Model (BEM). *Appl. Sci.* **2021**, *11*, 2167. [CrossRef]
47. Chen, S.; Jin, R.; Alam, M. Investigation of interoperability between building information modelling (BIM) and building energy simulation (BES). *Int. Rev. Appl. Sci. Eng.* **2018**, *9*, 137–144. [CrossRef]
48. Fernald, H.; Hong, S.; Bucking, S.; O'Brien, W. BIM to BEM translation workflows and their challenges a case study using a detailed BIM model. In Proceedings of the eSim2018, Montréal, QC, Canada, 9–10 May 2018.
49. gbXML Schema Validator. Available online: https://validator.gbxml.org (accessed on 11 December 2021).
50. Clipper: An Open Source Freeware Library for Clipping and Offsetting Lines and Polygons. Available online: http://www.angusj.com/delphi/clipper.php (accessed on 11 December 2021).
51. Gingerbread: A Lightweight gbXML Export Module for Revit. Available online: https://github.com/ian-quinn/gingerbread (accessed on 11 December 2021).
52. Aragog gbXML Viewer R14. Available online: https://www.ladybug.tools/spider/gbxml-viewer/r14/gv-cor-core/gv-cor.html (accessed on 11 December 2021).