Automatic generation of second-level space boundary topology from IFC geometry inputs

Abstract

The Industry Foundation Classes (IFC) is a semantically rich data model providing necessary information to support extraction of information necessary for the setup of building energy simulations. Often, 2^{nd}-level space boundary data contained in IFC, are missing or incorrect. To facilitate the connection between BIMs and energy simulation programs, the Common Boundary Intersection Projection (CBIP) algorithm is introduced. CBIP uses the geometric representations of building entities obtained from IFC files to generate the building's 2^{nd}-level space boundary topology. A prototypical implementation of the CBIP algorithm is used in a complex geometry building, as a verification of the capability of the algorithm to identify space boundaries.

Keywords: Industry Foundation Class, Building Information Model, Building Energy Performance Simulation, Second-level Space Boundaries

Introduction

The recent requirement for efficient allocation of energy resources in the building sector, has resulted in the increased use of building thermal simulations, during both the building design [1, 2] and operation phases [3, 4]. The accuracy of a thermal simulation model strongly depends on the accurate definition of building geometric characteristics, which include: the building envelope, the building orientation, the configuration of spaces, surfaces and volumes.

In current state of practice, it is quite common that a Computer-Aided Design (CAD) tool is used to represent the geometry. However, such an architectural perspective must be altered, in order for energy simulations to be performed [5]. Hence, building geometrical data extracted from CAD programs have to be manually transformed and combined with material properties to be entered as inputs to energy simulation routines, a process which is both time consuming and error-prone. CAD data have simple semantics, Building Information Models (BIM) [6] provide an improved way for information storage with richer semantics that include: building geometry, material data and information on building services. Open BIM data schemas include the Industry Foundation Classes (IFC) [7] and the green-building XML schema (gbXML) [8]. The popularity of these two open BIM schemas led many leading AEC software companies to implement support for gbXML- and IFC-based exchanges within their BIM authoring suites. Examples of such tools are Revit (AutoDesk) [9] and ArchiCad (GraphiSoft) [10].

A wide variety of promising attempts have been proposed to establish an automated data exchange between BIM and thermal simulation tools. The IDF Generator [11], developed at the Lawrence Berkeley National Laboratory (LBNL), works in conjunction with the Geometry Simplification Tool (GST) and transforms IFC-format building geometry into EnergyPlus input-data file (IDF) [12]; GST simplifies the original building geometry defined in IFC-format and converts it into gbXML-format, while the IDF Generator converts the gbXML-format file into EnergyPlus input-data file. The resulting IDF file contains all information related to building geometry and constructions needed to run an EnergyPlus simulation. IDF Generator is proposed as a semi-automated process, since for complex building geometries a manual manipulation of IDF geometry is required, including some corrections to windows in curtain walls, missing floors and ceilings [13]. The RIUSKA [14], developed by Granlund, uses the DOE-2.1 [15] thermal simulation engine and imports the building geometry from an IFC file, utilizing the BSPro server middleware ([16]). Limitations of its IFC import exist, since RIUSKA ignores slabs in the IFC file and simply generates them internally, according to the size of the space defined by the bounding walls. Moreover, high quality of import results are achieved only when RIUSKA is used in conjunction with SMOG, while compatibility problems occur when other CAD tools are used to author and export the IFC file.

In the AEC software industry, Green Building Studio (GBS) [17] web service uploads the geometry in gbXML-format and converts it into a DOE-2.2 or an EnergyPlus-format file. There are studies proving that several problems occur during the conversion process [18], including incorrect shading surface definitions and omission of some walls. Trimble's SketchUp together with its Openstudio and IFC2SKP plugins, is able to upload any gbXML or IFC well-formatted geometry and convert it into the EnergyPlus or TRNSYS17-format file [19]. However, lack of maturity of the import tool, neglects some information related to floors and ceilings. Virtual Environment (VE), developed by Integrated Environmental Solutions (IES) [20], is an integrated system that uses its own simulation engine, called Apache. IES VE supports import of gbXML and IFC file formats. Nevertheless, import results rely on the correctness of 2$^{\text{nd}}$-level space boundary geometry contained in IFC, which currently is not exported properly by any BIM authoring tool.

Among the two most popular BIM schemes, gbXML and IFC, IFC appears to be a suitable choice as its more rich in content, enables interoperability among different software environments and can be updated according to the building's modifications [21].

Concerning the building geometry, IFC can provide static building information that include geometric configuration and material properties, but in a form that might not be directly usable for the generation of thermal simulation models due to the absence of 2^{nd}-level space boundary information [13]. Hence, a consistent approach is required to extract building geometry information, contained in an IFC file, and subsequently to correctly identify the 2nd-level space boundary information.

In view of this, several algorithms have been proposed [22],[23], [24], which are based on graph theory and convert a three-dimensional architectural building model into the second-level space boundary topology without the need for definition of conditioned building space volumes.

In this work, following a different approach to address the 2nd-level space boundary generation requirement, the Common Boundary Intersection Projection (CBIP) algorithm is presented. A recent study has shown that thermal models obtained based on CBIP algorithm results, are comparable to models of other popular programs [25, 26].

CBIP algorithm can be applied to building geometries which do not contain design errors or building space incorrect definitions. In [27], errors that affect the creation of properly defined 2nd-level space bounaries are presented. Commercial software, such as Solibri Model Checker, [28] are able to identify such errors, which are communicated back to the AEC software and corrected manually. With an IFC free of design errors and building space incorrect definitions at hand, its geometric data can be used as input to CBIP algorithm.

Algorithmically, CBIP is divided into four operational stages: the Identification (ID) stage, the Boundary Surface Extraction (BSE) stage, the Common Boundary Intersection (CBI) stage and the Boundary Intersection Projection (BIP) stage which are analysed in Section 4. CBIP's stages involve geometric operations based on well-known methods for representing shapes, therefore an initial description of such methods, adopted by the algorithm, are presented in Sections 2 and 3. The output of the algorithm used to update the IFC database and the respective Space Boundary class, is described in Section 5, while design requirements and design recommendations to ensure the correct execution of the algorithm are discussed in Sections 6 and 7, respectively. Finally, CBIP has been tested on a demonstration building of a high geometry complexity and its results are presented in Section 9.

CBIP algorithm – Geometric Definitions

CBIP takes as input the geometric representations of various building entities, which are assumed to be polyhedrons, performs certain operations on them and outputs polygonal surfaces which are the 2nd-level space boundaries. Consequently, CBIP algorithm's mathematical foundation consists of geometric operations, applied on geometric representations of the involved building entities.

Various geometric representation methods including the octree and the Boundary representation have been used in Building Information Models [29]. In CBIP two such methods are used. The first, is the Boundary representation (B-rep) [30], described in Section 2.1. B-rep theory is adopted in order to describe each polyhedron by its corresponding boundary polygons. Additionally, to determine the space boundaries which are essentially common surfaces shared by two polyhedrons, the Binary Space Partitioning tree (BSP-tree) polyhedral representation [31] is adopted and described in Section 2.2.

Boundary representation

The B-rep of a polyhedron $\mathbf{A}$ associated with a building entity, is denoted by $\partial\mathbf{A}$. Essentially, $\partial\mathbf{A}$ is a set of boundary polygon surfaces $\partial\mathbf{A} = \{A_1, ..., A_i, ..., A_N\}$ (see figure 1). Each boundary polygon surface in this representation, conforms to the right hand outward normal convention: the direction of the normal vector $\hat{n}_{A_i}$ of every boundary polygon A_i evaluated using the right hand, is towards the exterior of the polyhedron $\mathbf{A}$, as displayed in figure 1. The right hand normal vector direction evaluation method proceeds as follows: when the fingers of the right hand, excluding the thumb, follow the points of the polygon A_i, the thumb points to the direction of the normal vector. Consequently, the outward normal convention, with the direction of the normal vectors evaluated using right hand, requires the boundary polygon points to be correctly ordered: in a counter clock-wise manner when looking from outside the polyhedron (as displayed in figure 1).

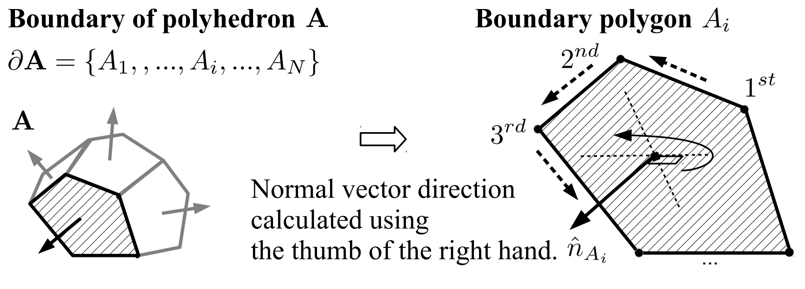

Boundary of polyhedron A
$\partial\mathbf{A} = \{A_1, , ..., A_i, ..., A_N\}$
A
Boundary polygon A_i
2^{nd}
1^{st}
3^{rd}
Normal vector direction calculated using the thumb of the right hand. $\hat{n}_{A_i}$
...

Figure 1: Boundary representation (B-rep) of a polyhedron as a set of boundary polygon surfaces $\partial\mathbf{A} = \{A_1, ..., A_i, ..., A_N\}$. When the points of each polygon surface A_i are correctly ordered (counter clockwise when looking from outside the polyhedron) the direction of the normal vector of the surface $\hat{n}_{A_i}$, evaluated using the right hand, is towards the exterior of the polyhedron.

Binary Space Partitioning tree representation

A Binary Space Partitioning tree representation or BSP-tree refer generally to a set of surfaces $\mathcal{A}$ and is denoted by $T_{\mathcal{A}}$. Since most of the surface sets encountered here are B-reps, the respective BSP-trees refer to polyhedrons $\mathbf{A}$ and are denoted by $T_{\mathbf{A}}$, instead of $T_{\partial\mathbf{A}}$ for simplicity.

An example of a BSP-tree referring to an intersection surface set $\mathcal{A}_I \subset \partial\mathbf{A}$ of a polyhedron $\mathbf{A}$, is displayed in figure 2. In a broad sense, $T_{\mathbf{A}}$ contains the boundary polygons of $\partial\mathbf{A}$ and defines a partition of the 3D space into a finite set of sub-spaces, depending on the orientation of the polyhedron's boundary surfaces. In a broad sense, a BSP-tree of a polyhedron is a binary tree that partitions the 3D space into finite number of sub-spaces according to the outward normal vectors of its boundary surfaces, as described below.

$T_{\mathbf{A}}$ is a structure with three fields. The root value of a BSP-tree $T_{\mathbf{A}}$ contains a single-root polygon, or multiple-root coplanar polygons with identical normal vectors, and is denoted by the *pol* field ($T_{\mathbf{A}}.pol$). The plane of the root divides the 3D space into two sub-spaces, the outside and the inside sub-space. The outside sub-space is indicated by the common normal vector of the root polygon(s). The outside sub-space contains polygons which are placed in the right sub-tree of $T_{\mathbf{A}}$ and is denoted by the field $T_{\mathbf{A}}.out$. The inside sub-space is indicated by the opposite of the normal vector of the root polygon(s). The inside sub-space contains polygons which are placed in the left sub-tree of $T_{\mathbf{A}}$, and is denoted by the field $T_{\mathbf{A}}.ins$.

Consequently, moving from the root to the leaves of the tree and following the left/right branches lead to inside/outside sub-spaces (opposite/towards the direction of the outward normal vectors), respectively. The final partitions (sub-regions) of the 3D space are indicated by the leaves of the tree which contain binary values. By convention, a leaf has value 1, if the respective sub-region is inside the polygons of the node above the leaf, and the value 0, if the respective subregion lies outside these polygons.

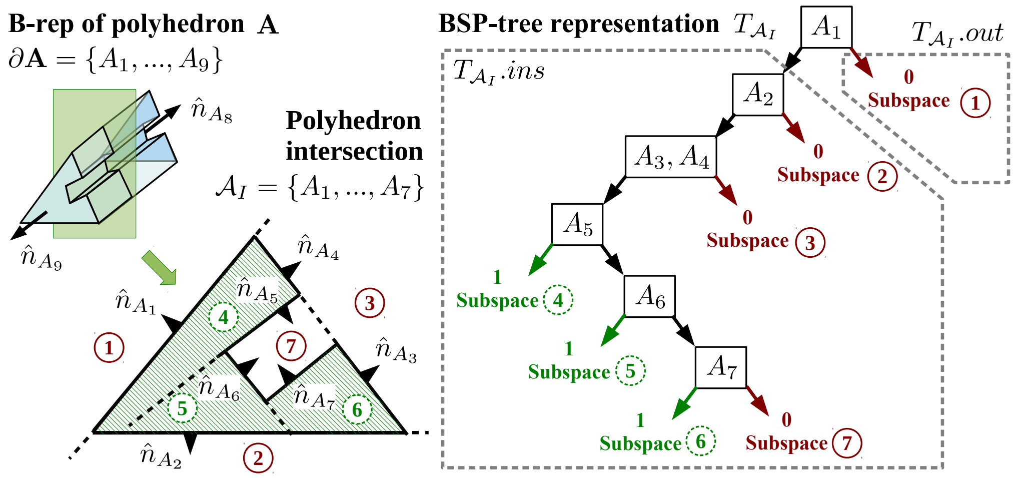


Figure 2: BSP-tree representation of an intersection surface set $\mathcal{A}_I = \{A_1, ..., A_7\} \subset \partial\mathbf{A}$ of a polyhedron $\mathbf{A}$ creating a non-convex region.

If two boundary polygons are coplanar but their outward normal vectors have opposite directions, one polygon is considered to lie outside the other and therefore are placed in separate root nodes. The sequence of the boundary polygons of $\partial\mathbf{A}$, used to populate the BSP-tree $T_{\mathbf{A}}$, does not matter. The tree representation $T_{\mathbf{A}}$ of a polyhedron $\mathbf{A}$ is obtained from its B-rep representation $\partial\mathbf{A}$ using the recursive algorithm described in [31].

CBIP algorithm – Geometric operations

To obtain the 2nd-level space boundary surfaces, which are parts of common boundary surfaces between the polyhedral representations of building spaces and the polyhedral representations of building constructions, CBIP performs geometric operations defined by three geometric clipping functions. These clipping functions are applied on polyhedral pairs $\mathbf{A}$, $\mathbf{B}$, and use: (1) their B-reps $\partial\mathbf{A} = \{A_1, ..., A_{N_{\partial\mathbf{A}}}\}$, $\partial\mathbf{B} = \{B_1, ..., B_{N_{\partial\mathbf{B}}}\}$, with $N_{\partial\mathbf{A}}$, $N_{\partial\mathbf{B}}$ the cardinalities of the sets $\partial\mathbf{A}$, $\partial\mathbf{B}$; (2) the respective BSP-tree representations $T_{\mathbf{A}}$, $T_{\mathbf{B}}$; (3) two polygon clipping operators c_1 and c_2; and (4) a polygon set partition function.

Polygon clipping operators

Polygon clipping operators c_1 and c_2 involve two polygons A_i and B_j. Essentially, c_1 and c_2 modify their second operand (polygon A_i), depending on the relative position of their first operand (polygon B_j) and the direction of its normal vector $\hat{n}_{B_j}$ (see figure 3).

Mathematically, these operations are defined by:

$$A_{1i} = B_j(c_1)A_i \text{ and } A_{2i} = B_j(c_2)A_i \qquad (1)$$

Generally, three clipping cases can be distinguished:

A. The plane of B_j *dissects* A_i into two parts: A_{1i} towards the normal vector $\hat{n}_{B_j}$ and A_{2i} towards the opposite direction $-\hat{n}_{B_j}$. This dissection is performed by c_1 or c_2, returning A_{1i} or A_{2i}, respectively ($A_i = A_{1i} \cup A_{2i}$).

B1. The plane of B_j *dissects the plane of* A_i into two half-planes and A_i is in the half-space pointed by $\hat{n}_{B_j}$. In this case c_1 returns A_i and c_2 returns an empty set.

B2. The plane of B_j *dissects the plane of* A_i into two half-planes and A_i is in the half-space pointed by $-\hat{n}_{B_j}$. In this case c_1 returns an empty set and c_2 returns A_i.

The previous clipping operations are implemented using 2D set operations on polygons (intersection, union and subtraction), following the algorithm proposed in [32].

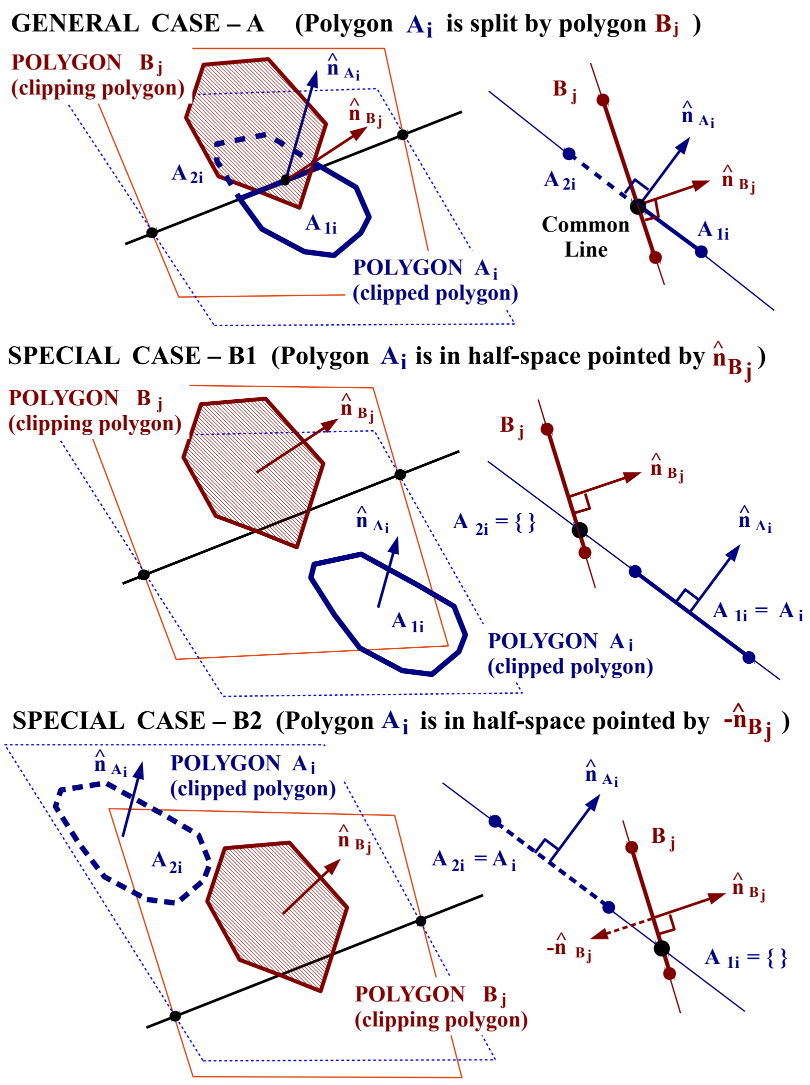
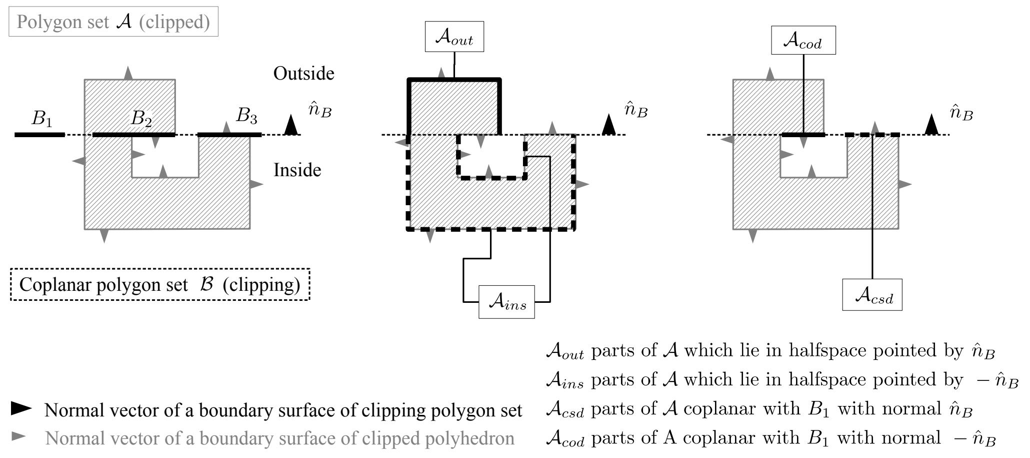

GENERAL CASE – A (Polygon A_i is split by polygon B_j)
POLYGON B_j (clipping polygon)
n̂ A_i
n̂ B_j
A 2i
A 1i
POLYGON A_i (clipped polygon)
B_j
Common Line
SPECIAL CASE – B1 (Polygon A_i is in half-space pointed by n̂ B_j)
A 2i = { }
A 1i = A_i
SPECIAL CASE – B2 (Polygon A_i is in half-space pointed by -n̂ B_j)
A 2i = A_i
-n̂ B_j
A 1i = { }

Figure 3: Illustration of polygon clipping operators c_1 and c_2 on polygon A_i by polygon B_j.

Polygon set partition function

The polygon set partition function P, used by the geometric clipping functions, can be defined as a partition of a polygon set $\mathcal{A}$, intersected by a set of coplanar polygons (partition set $\mathcal{B}$) with the same outward normal vector $\hat{n}_B$ (see figure 4).



Polygon set $\mathcal{A}$ (clipped)
Outside
Inside
B_1
B_2
B_3
$\hat{n}_B$
$\mathcal{A}_{out}$
$\mathcal{A}_{ins}$
$\mathcal{A}_{cod}$
$\mathcal{A}_{csd}$
Coplanar polygon set $\mathcal{B}$ (clipping)
Normal vector of a boundary surface of clipping polygon set
Normal vector of a boundary surface of clipped polyhedron
$\mathcal{A}_{out}$ parts of $\mathcal{A}$ which lie in halfspace pointed by $\hat{n}_B$
$\mathcal{A}_{ins}$ parts of $\mathcal{A}$ which lie in halfspace pointed by $-\hat{n}_B$
$\mathcal{A}_{csd}$ parts of $\mathcal{A}$ coplanar with B_1 with normal $\hat{n}_B$
$\mathcal{A}_{cod}$ parts of A coplanar with B_1 with normal $-\hat{n}_B$

Figure 4: Illustration of the polygon set partition function

Mathematically, the polygon set partition function P is defined by the following expression:

$$[\mathcal{A}_{ins},\ \mathcal{A}_{csd},\ \mathcal{A}_{cod},\ \mathcal{A}_{out}] = P(\mathcal{B},\mathcal{A}) \tag{2}$$

The returning arguments $\mathcal{A}_{ins}$ and $\mathcal{A}_{out}$ are subsets of the set $\mathcal{A}$ containing polygons lying in the half-space pointed by $-\hat{n}_B$ and $\hat{n}_B$, respectively. $\mathcal{A}_{cop}$ and $\mathcal{A}_{csd}$ contain polygons coplanar with the polygons in $\mathcal{B}$, which have opposite (cod) and same direction (csd) normals with $\hat{n}_B$, respectively (see figure 4). The above sets are populated using Algorithm 1 and the polygon clipping operators c_1 and c_2.

Polygon set clipping functions

Using the operators c_1, c_2, and the polygon partition function P, three recursive clipping functions F_{ins}, F_{out} and F_{cod} are defined. These functions are applied on a polygon set $\mathcal{A}$ (clipped polyhedron), using the BSP-tree representation $T_{\mathbf{B}}$ of a polyhedron $\mathbf{B}$ (clipping polyhedron). These clipping functions return:

Algorithm 1 *Partition function $P(\mathcal{B},\mathcal{A})$*

```
$\mathcal{A} = \{A_1, ..., A_N\}$                                   // Polygon set (to be partitioned) //
$\mathcal{B} = \{B_1, ..., B_M\}$                                   // Polygon set (partitioning set) //
$\mathcal{A}_{ins} = \emptyset$, $\mathcal{A}_{csd} = \emptyset$, $\mathcal{A}_{cod} = \emptyset$, $\mathcal{A}_{out} = \emptyset$             // Initialize output sets //
for $i = 1 : N$ do
    if $A_i \in \mathcal{A}, B_1$ are coplanar then
        for $j = 1 : M$ do
            $A_i \leftarrow A_i - B_j$                       // Subtract $B_j$ from $A_i$ and update $A_i$ //
            $AIB_{ij} = A_i \cap B_j$             // Intersect $B_j$ with $A_i$ and form $AIB_{ij}$ polygon //
            if $\hat{n}_{A_i} \uparrow\uparrow \hat{n}_{B_j}$ then
                $\mathcal{A}_{csd} \leftarrow \mathcal{A}_{csd} \cup AIB_{ij}$            // Include polygon $AIB_{ij}$ in $\mathcal{A}_{csd}$ set //
            else
                $\mathcal{A}_{cod} \leftarrow \mathcal{A}_{cod} \cup AIB_{ij}$            // Include polygon $AIB_{ij}$ in $\mathcal{A}_{cod}$ set //
            end if
        end for
        $\mathcal{A}_{out} \leftarrow \mathcal{A}_{out} \cup A_i$                        // Include polygon $A_i$ in $\mathcal{A}_{out}$ set //
    else
        $\mathcal{A}_{ins} \leftarrow \mathcal{A}_{ins} \cup [B_1(c_2)A_i]$     // Include clipped polygon $[B_1(c_2)A_i]$ in $\mathcal{A}_{ins}$ set //
        $\mathcal{A}_{out} \leftarrow \mathcal{A}_{out} \cup [B_1(c_1)A_i]$     // Include clipped polygon $[B_1(c_1)A_i]$ in $\mathcal{A}_{out}$ set //
    end if
end for
```

- $\mathcal{A}_{ins} = F_{ins}(T_{\mathbf{B}}, \mathcal{A})$: The parts of $\mathcal{A}$, which are inside polyhedron $\mathbf{B}$;
- $\mathcal{A}_{out} = F_{out}(T_{\mathbf{B}}, \mathcal{A})$: The parts of $\mathcal{A}$, which are outside polyhedron $\mathbf{B}$;
- $\mathcal{A}_{cod} = F_{cod}(T_{\mathbf{B}}, \mathcal{A})$: The parts of $\mathcal{A}$, which are coplanar with the surfaces of $\partial\mathbf{B}$ and have opposite outward normal vectors.

Function F_{ins} is described by Algorithm 2, function F_{out} by Algorithm 3 and function F_{cod} by Algorithm 4. In these algorithms, the clipping BSP-tree $T_{\mathbf{B}}$ has three fields: $T_{\mathbf{B}}.pol$ refers to the polygons contained in the root of $T_{\mathbf{B}}$; $T_{\mathbf{B}}.ins$ contains the left (inside) sub-tree of $T_{\mathbf{B}}$; and $T_{\mathbf{B}}.out$ contains the right (outside) sub-tree of $T_{\mathbf{B}}$.

The F_{cod} function is used by CBIP to identify the Common Boundary Intersection surfaces, which are coplanar surface pairs belonging to two different polyhedrons and have opposite normal vectors. Examples of the clipping functions F_{out}, F_{ins} and F_{cod}, applied on $\mathcal{A}$, using a clipping polyhedron $\mathbf{B}$ are displayed in figure 5.

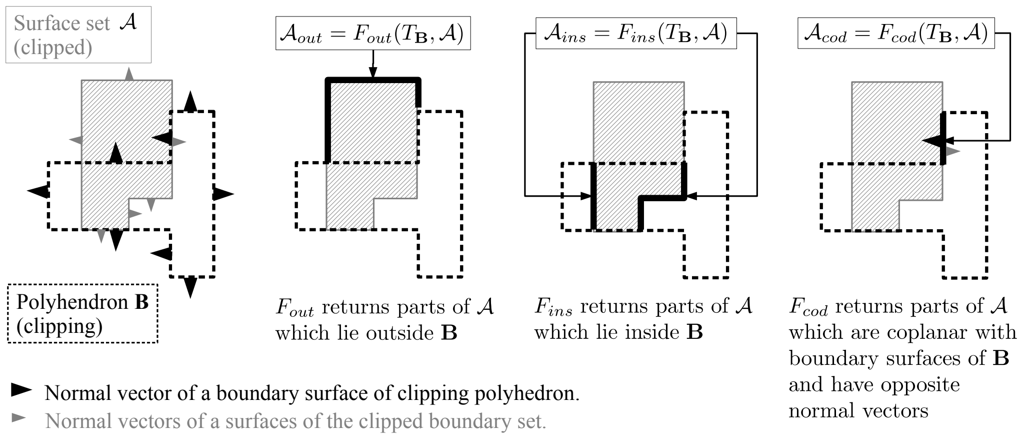

Surface set $\mathcal{A}$ (clipped)
Polyhendron $\mathbf{B}$ (clipping)
$\mathcal{A}_{out} = F_{out}(T_{\mathbf{B}}, \mathcal{A})$
$\mathcal{A}_{ins} = F_{ins}(T_{\mathbf{B}}, \mathcal{A})$
$\mathcal{A}_{cod} = F_{cod}(T_{\mathbf{B}}, \mathcal{A})$
F_{out} returns parts of $\mathcal{A}$ which lie outside $\mathbf{B}$
F_{ins} returns parts of $\mathcal{A}$ which lie inside $\mathbf{B}$
F_{cod} returns parts of $\mathcal{A}$ which are coplanar with boundary surfaces of $\mathbf{B}$ and have opposite normal vectors
Normal vector of a boundary surface of clipping polyhedron.
Normal vectors of a surfaces of the clipped boundary set.

Figure 5: Results of clipping functions – $\mathcal{A}$ is the polygon set of a clipped polyhedron and $\mathbf{B}$ is the clipping polyhedron

Algorithm 2 *Inside clipping function F_{ins}:* $\mathcal{A}_{ins} = F_{ins}(T_{\mathbf{B}}, \mathcal{A})$

```
if $T_{\mathbf{B}}$ is binary then
    if $T_{\mathbf{B}} = 0$ then
        $\mathcal{A}_{ins} = \emptyset$                                   // Initialize output set $\mathcal{A}_{ins}$ //
    end if
    if $T_{\mathbf{B}} = 1$ then
        $\mathcal{A}_{ins} = \mathcal{A}$                                 // Initialize output set $\mathcal{A}_{ins}$ with $\mathcal{A}$ //
    end if
else
    $[\mathcal{A}_{ins},\ \mathcal{A}_{csd},\ \mathcal{A}_{cod},\ \mathcal{A}_{out}] = P(T_{\mathbf{B}}.pol, \mathcal{A})$      // Partition $\mathcal{A}$ with $T_{\mathbf{B}}.pol$ //
    if $\mathcal{A}_{ins} \neq \emptyset$ then
        $\mathcal{A}_{i,ins} = F_{ins}(T_{\mathbf{B}}.ins, \mathcal{A}_{ins})$      // Apply $F_{ins}$ recursively on $\mathcal{A}_{ins}$ with $T_{\mathbf{B}}.ins$ //
        $\mathcal{A}_{ins} \leftarrow \mathcal{A}_{ins} \cup \mathcal{A}_{i,ins}$                  // Include $\mathcal{A}_{i,ins}$ in $\mathcal{A}_{ins}$ //
    end if
    if $\mathcal{A}_{out} \neq \emptyset$ then
        $\mathcal{A}_{i,out} = F_{ins}(T_{\mathbf{B}}.out, \mathcal{A}_{out})$      // Apply $F_{ins}$ recursively on $\mathcal{A}_{out}$ with $T_{\mathbf{B}}.out$ //
        $\mathcal{A}_{ins} \leftarrow \mathcal{A}_{ins} \cup \mathcal{A}_{i,out}$                  // Include $\mathcal{A}_{i,out}$ in $\mathcal{A}_{ins}$ //
    end if
end if
```

Algorithm 3 *Outside clipping function* F_{out}: $\mathcal{A}_{out} = F_{out}(T_{\mathbf{B}}, \mathcal{A})$

```
if $T_{\mathbf{B}}$ is binary then
    if $T_{\mathbf{B}} = 0$ then
        $\mathcal{A}_{out} = \mathcal{A}$                          // Initialize output set $\mathcal{A}_{out}$ with $\mathcal{A}$ //
    end if
    if $T_{\mathbf{B}} = 1$ then
        $\mathcal{A}_{out} = \emptyset$                                  // Initialize output set $\mathcal{A}_{out}$ //
    end if
else
    $[\mathcal{A}_{ins},\ \mathcal{A}_{csd},\ \mathcal{A}_{cod},\ \mathcal{A}_{out}] = P(T_{\mathbf{B}}.pol, \mathcal{A})$       // Partition $\mathcal{A}$ with $T_{\mathbf{B}}.pol$ //
    if $\mathcal{A}_{ins} \neq \emptyset$ then
        $\mathcal{A}_{o,ins} = F_{out}(T_{\mathbf{B}}.ins, \mathcal{A}_{ins})$    // Apply $F_{out}$ recursively on $\mathcal{A}_{ins}$ with $T_{\mathbf{B}}.ins$ //
        $\mathcal{A}_{out} \leftarrow \mathcal{A}_{out} \cup \mathcal{A}_{o,ins}$                      // Include $\mathcal{A}_{o,ins}$ in $\mathcal{A}_{out}$ //
    end if
    if $\mathcal{A}_{out} \neq \emptyset$ then
        $\mathcal{A}_{o,out} = F_{out}(T_{\mathbf{B}}.out, \mathcal{A}_{out})$    // Apply $F_{out}$ recursively on $\mathcal{A}_{out}$ with $T_{\mathbf{B}}.out$ //
        $\mathcal{A}_{out} \leftarrow \mathcal{A}_{out} \cup \mathcal{A}_{o,out}$                      // Include $\mathcal{A}_{o,out}$ in $\mathcal{A}_{out}$ //
    end if
end if
```

Algorithm 4 *Coplanar opposite direction clipping function*
F_{cod}: $\mathcal{A}_{cod} = F_{cod}(T_{\mathbf{B}}, \mathcal{A})$

```
if $T_{\mathbf{B}}$ is a tree (not binary value) then
    $[\mathcal{A}_{ins},\ \mathcal{A}_{csd},\ \mathcal{A}_{cod}\ \ \mathcal{A}_{out}] = P(T_{\mathbf{B}}.pol, \mathcal{A})$       // Partition $\mathcal{A}$ with $T_{\mathbf{B}}.pol$ //
    if $\mathcal{A}_{cod} \neq \emptyset$ then
        $\mathcal{A}_{cod} \leftarrow \mathcal{A}_{cod}$                            // Initialize output set $\mathcal{A}_{cod}$ //
    end if
    if $\mathcal{A}_{ins} \neq \emptyset$ then
        $\mathcal{A}_{c,ins} = F_{cod}(T_{\mathbf{B}}.ins, \mathcal{A}_{ins})$    // Apply $F_{cod}$ recursively on $\mathcal{A}_{ins}$ with $T_{\mathbf{B}}.ins$ //
        $\mathcal{A}_{cod} \leftarrow \mathcal{A}_{cod} \cup \mathcal{A}_{c,ins}$                      // Include $\mathcal{A}_{c,ins}$ in $\mathcal{A}_{cod}$ //
    end if
    if $\mathcal{A}_{out} \neq \emptyset$ then
        $\mathcal{A}_{c,out} = F_{cod}(T_{\mathbf{B}}.out, \mathcal{A}_{out})$    // Apply $F_{cod}$ recursively on $\mathcal{A}_{out}$ with $T_{\mathbf{B}}.out$ //
        $\mathcal{A}_{cod} \leftarrow \mathcal{A}_{cod} \cup \mathcal{A}_{c,out}$                      // Include $\mathcal{A}_{c,out}$ in $\mathcal{A}_{cod}$ //
    end if
end if
```

CBIP algorithm stages

As mentioned earlier, CBIP consists of four operational stages. CBIP's input contains IFC geometric data related to three types of building entities: Constructions, Openings and Volumes. The final output of the CBIP process is the generation of the 2$^{\text{nd}}$-level space boundaries, which are essentially surface pairs, associated with four types of thermal simulation elements.

The input data of CBIP are gathered in the first stage. Their classification to Constructions, Openings and Volumes is performed according to their roles in a thermal simulation process. The first stage is described in Section 4.1.

The scope of the second stage, is to generate the B-reps of the building entities, isolated from the first stage. This is accomplished using a process called *Boundary Surface Extraction (BSE)*, described in Section 4.2. In some cases, building entities of the Construction type may contain entities of the Opening type, for instance building walls (Constructions) which contain doors or windows. In such cases, the B-reps of these constructions have to be updated by subtracting the B-reps of the opening volumes, they contain. This is performed by the *Opening Construction Subtraction (OCS)* process, described in Section 4.2.1.

The boundary surfaces' B-reps, deduced from the second stage, are processed further in the third stage, where the *Common Boundary Intersection (CBI)* process (described in Section 4.3) is applied to obtain the Common Boundary (CB) surfaces shared by B-rep pairs. CB surfaces' types are denoted as *Primary types* and are described in Section 4.3.1. The remaining B-rep surfaces, which are not CB surfaces, are also gathered using the *Remaining Surface Extraction (RSE)* process (described in Section 4.3.2), and are marked as *Derived types* of surfaces, which are attached to the environment.

Finally, the 2nd-level space boundary surfaces, the associated four types of thermal simulation model elements (thermal, shades, openings and air boundaries) and their connectivity information are obtained in the fourth stage. This is accomplished by projection of a CB surface (first surface), obtained from the third stage, to the plane of another CB surface (second surface) and vice versa. This process, called *Boundary Intersection Projection (BIP)*, is described in Section 4.4.

Identification stage - ID (stage 1)

Even though IFC files contain information referring to multiple building geometry entities, only some of them are required for building thermal simulations. These building geometry entities can be classified into three categories depending on their role in thermal building simulations: Constructions, Openings and Volumes.

Constructions are single- or multi-layer entities, which are involved in thermal simulations in two different ways: (1) directly, by impeding thermal energy flow between building volumes, where the construction layers and their specific thermal properties are taken into account; and (2) indirectly, by blocking sunlight, thus impeding solar heat gains (shading), where their thermal behavior is not considered. Certain IFC classes, which refer to building constructions, belong to the abstract *IfcBuildingElement* class and are indicated by the "CONSTRUCTIONS" dashed rectangle in figure 7.

Openings are building entities described by the *IfcOpening* class. These entities contain doors, windows and skylights, which are generally holes on building Constructions. These entities play important role in thermal simulations, since depending on their state either impede or allow thermal flow. The *IfcOpeningElement* class contains information associated with building openings and belongs to the abstract *IfcElement* class. This class and its relations are indicated by the "OPENINGS" dashed rectangle in figure 7.

Building volumes are entities that exchange thermal energy, which are categorized as follows:

- *Building spaces* refer to the air volumes of rooms or room partitions (separated by air boundaries). Building spaces interchange thermal energy with other spaces, with the surrounding environment or with the site encompassing the building. Building spaces are defined by the *IfcSpace* class.

- *Building site* refers to the surrounding ground volume, encompassing the building under consideration. The building site is defined by the *IfcSite* class.

IFC classes related to Volume entities, belong to the abstract *IfcBuildingSpatialStructureElement* class and are indicated by the "VOLUMES" dashed rectangle in figure 7.

The aforementioned building entities are extracted and their polyhedral boundary surface representations are obtained from their boundary surfaces, as described in Section 4.2.

Boundary Surface Extraction stage - BSE (stage 2)

In IFC, all relative building entities, required for the execution of CBIP, are considered products which are related to the abstract *IfcProduct* class. All associated products have a 3D shape representation, condition which is met by the Design Transfer View definition [33], as figure 7 indicates. However, an essential input requirement of CBIP algorithm, is that all involved products must have an outward oriented boundary surface geometric representation (B-rep), as described in Section 2.1, condition which is not always satisfied. Hence, further processing on some products' shape representations is required to obtain the desired B-reps. The required data for the generation of the B-reps are contained in the *IfcGeometricRepresentationItem* class, related to the *IfcProductDefinitionShape* subclass of the *IfcProduct* class (see figure 7).

There are five main solid geometrical representations and respective sub-classes of the *IfcGeometricRepresentationItem* class, according to figure 7. The involved geometric representations and the respective IFC classes, contained in Design Transfer View 1.0 [33], are:

(1) *Face based surface model representation*, described by *IfcBasedSurfaceModel* class – According to this representation, the solid of the building entity is described by a set of boundary surfaces "faces" in a 3D space. Such representation need no further processing and can be used directly by the CBIP algorithm, provided that the surfaces are correctly oriented.

(2) *Solid model representation*, described by *IfcSolidModel* class – This class consists of five subclasses referring to the way the solid model is being represented:

- *Manifold solid representation*, described by *IfcManifoldSolidBrep* class – A manifold solid B-rep is a finite, arc-wise connected volume bounded by one or more surfaces, each of which is a connected, oriented, finite, closed 2D-manifold. In this case no further processing is required, since all the points of the boundary surfaces are given.
- *Swept area solid representation*, described by *IfcSweptAreaSolid* class – This class contains solids, either described by a 2D profile being extruded towards a given direction and length (*IfcExtrudedAreaSolid*), or revolved around a fixed axes (*IfcRevolvedAreaSolid*), or translated along a curve trajectory (*IfcSurfaceCurvedSweptAreaSolid*).

  In this case, based on the base profile points, the extrusion direction and the extrusion length, the remaining points of the boundary surfaces are calculated and the respective boundary polygons are obtained. Essentially, the obtained base points are being translated or rotated (depending on the case) following a certain direction, generating the rest boundary surface points.

(3) *Half Space Solid representation*, described by *IfcHalfSpaceSolid* class – Two cases of half-space solid representation can be distinguished:

- *Polygonal bounded half-space* representation – the half space solid is bounded by a base polygon that is extruded at a specific depth and is intersected by a 3D surface (plane or curved surface in general). As in the case of the extruded area solid, the points of the boundary surfaces are obtained from the base points, the extrusion direction and length, and the intersecting surface.
- *Boxed half-space* representation – similarly to the polygonal bounded half-space solid, it is bounded by a bounding box. In this case the points of the bounding box determine the points of the intersecting boundary surfaces.

(4) *Boolean result representation*, described by *IfcBooleanResult* class – This class refers to solid geometric representations, which are obtained by performing boolean operations (union, intersection, difference) on solids, represented by the previous classes. Consequently the B-reps of all the involved solids are extracted and the final results are obtained by the clipping functions, applied on the extracted B-reps.

Representations that do not contain the desired B-reps for CBIP, require geometric calculations. These calculations are preformed in the second stage of the *Boundary Surface Extraction (BSE)* process. The sub-classes, data of which require geometric calculations to obtain the respective B-reps, are indicated by dashed blocks in the Express-G diagram of figure 7.

Opening Construction Subtraction process (OCS)

Constructions containing openings are represented in IFC files as solid objects, without considering the openings as holes. Therefore, to obtain a more accurate B-rep of these constructions and to determine the common boundary surfaces among these constructions and their opening volumes (frames of doors, windows, etc.), the polyhedral geometrical representations of the opening volumes must be subtracted from the polyhedral representations of the constructions. Such subtraction is performed by the *Opening Construction Subtraction* (OCS) process, which uses the F_{ins} and F_{out} clipping functions, given as inputs: the B-rep $\partial\mathbf{A}$, the BSP-tree representation T_A of the construction $\mathbf{A}$, the B-rep $\partial\mathbf{A}_{op}$ and BSP-tree representation $T_{\mathbf{A}_{op}}$ of the union of its openings $\partial\mathbf{A}_{op}$: $\partial\mathbf{A}_{op} = \partial\mathbf{O}_1 \cup ... \cup \partial\mathbf{O}_N$ ($\partial\mathbf{O}_i$ is the B-rep of opening i).

OCS process returns a set of boundary polygons (B-rep) of the construction with its openings subtracted. OCS is illustrated in figure 6, for the case of a wall containing a door and a window.

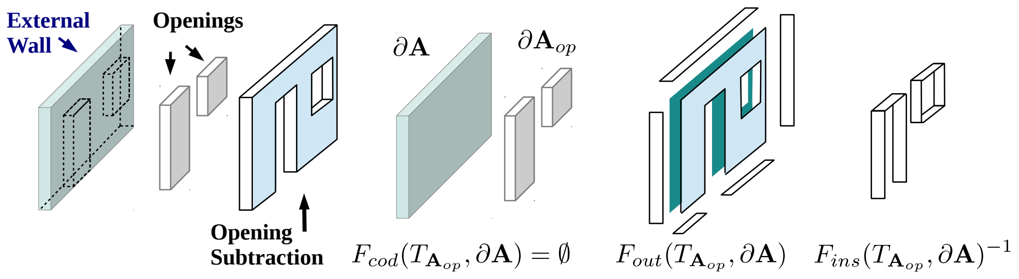

External Wall
Openings
Opening Subtraction
$\partial\mathbf{A}$
$\partial\mathbf{A}_{op}$
$F_{cod}(T_{\mathbf{A}_{op}}, \partial\mathbf{A}) = \emptyset$
$F_{out}(T_{\mathbf{A}_{op}}, \partial\mathbf{A})$
$F_{ins}(T_{\mathbf{A}_{op}}, \partial\mathbf{A})^{-1}$

Figure 6: Illustration of OCS process applied on a rectangular, wall containing door and window openings

Mathematically, OCS process is described as follows:

$$OCS(T_{\mathbf{A}}, \partial\mathbf{A}, T_{\mathbf{A}_{op}}, \partial\mathbf{A}_{op}) = \bigcup \begin{bmatrix} F_{out}(T_{\mathbf{A}_{op}}, \partial\mathbf{A}) \\ F_{cod}(T_{\mathbf{A}_{op}}, \partial\mathbf{A}) \\ F_{ins}(T_{\mathbf{A}}, \partial\mathbf{A}_{op})^{-1} \end{bmatrix} \tag{3}$$

The exponent -1, applied to F_{ins} function, inverts the ordering of the points of the obtained polygons, which also inverts their normal vectors.

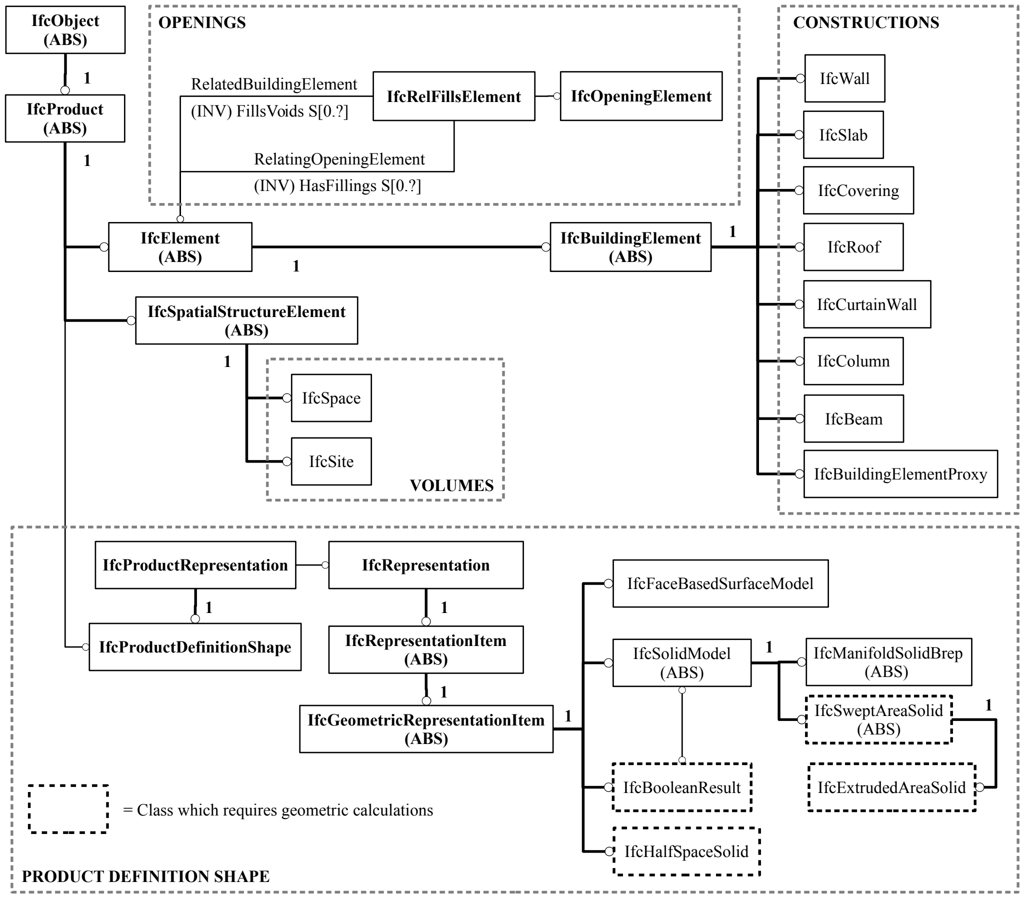


Figure 7: Part of IFC Design Transfer View 1.0 EXPRESS-G schema [33] containing the required classes for CBIP algorithm. Sub-classes which require calculations (performed by BSE) are indicated by dashed blocks.

Common Boundary Intersection stage - CBI (stage 3)

The Common Boundary Intersection (CBI) process determines the CB surfaces shared by two polyhedrons **A** and **B** representing two building entities. There are two types of CB surfaces: the *Primary type*, described in Section 4.3.1, and the *Derived type*, described in Section 4.3.2. In a nutshell, CBI is applied on the pairs $\partial\mathbf{A}$ and $\partial\mathbf{B}$ of polyhedrons **A** and **B**, and outputs the set of CB surfaces $\mathcal{CB}_{AB}$, shared by the two polyhedrons. After the opening volumes subtraction from their constructions, the CB surface set $\mathcal{CB}_{AB}$ is obtained by applying the F_{cod} clipping function on $\partial\mathbf{A}$ using the BSP-tree $T_{\mathcal{B}}$. CBI process is expressed mathematically by Equation 4.

$$\mathcal{CB}_{AB} = F_{cod}(T_{\mathbf{B}}, \partial\mathbf{A}) \tag{4}$$

Primary types of common boundary surfaces

After the OCS process, B-reps of the resulting building constructions (obtained from the OCS process) are forwarded to the CBI process, from where the following five primary types of CB surfaces, depicted in figure 8, are derived:

(1) *Construction - Construction (C-C)* CB surfaces. C-C CB surfaces are surfaces where constructions (walls, slabs, roofs, ...) touch other constructions. Although C-C surfaces are not used directly as elements of thermal models, they contribute towards specifying the construction - environment boundaries.

(2) *Construction - Volume (C-V)* CB surfaces. Examples of C-V CB surfaces include surfaces shared by walls and spaces, slabs and spaces, or slab and sites.

(3) *Volume - Volume (V-V)* CB surfaces. Examples of V-V CB surfaces include boundaries between building spaces and boundaries between building spaces and building site. Such boundaries do not impede the thermal energy flow among the building volumes.

(4) *Opening - Construction (O-C)* CB surfaces. Examples of O-C CB boundaries include the door and window frames and thresholds. Although such boundaries do not participate directly in the calculation of the thermal model elements, they contribute towards deriving the Opening-Environment (O-E) surfaces.

(5) *Opening - Volume (O-V)* CB surfaces. O-V CB boundaries include surfaces shared by openings and spaces, or openings and site. These surfaces contribute to derive the opening thermal simulation elements.

Derived types of surfaces (Environment surfaces)

After subtracting the Primary types of CB surfaces from the B-reps of the building entities, the remaining surfaces define surfaces attached to the environment. These surfaces are obtained by the *Remaining Surfaces Extraction (RSE)* process (see Algorithm 5).

Depending on the building entity's type (Construction, Opening or Volume), three sets of Derived (or environment) surfaces are defined (examples displayed in figure 8):

(1) *Construction - Environment (C-E)* CB surfaces. Examples of such surfaces include the external surfaces of a wall or a slab (balcony), attached to the outside air.

(2) *Opening - Environment (O-E)* CB surfaces. Examples of O-E surfaces include the external surfaces of doors and windows, attached to the outside air.

(3) *Volume - Environment (V-E)* CB surfaces. Examples of such surfaces include the external surfaces of spaces (flats), attached to the outside air.

Algorithm 5 *Remaining Surface Extraction (RSE)* process

```
$\mathbf{A}$                                   // Polyhedron under consideration //
$\partial\mathbf{A} = \{A_1, A_2, ..., A_N\}$                        // Polyhedral boundary //
$\mathcal{CB} = \{CB_1, ..., CB_M\}$          // Common boundaries of A and other polyhedrons //
$\mathcal{R} = \emptyset$                                // Initialization of remaining surface set //
for $i = 1 : N$ do
  for $j = 1 : M$ do
    if $A_i$ and $CB_j$ are coplanar. then
      $A_i \leftarrow A_i - CB_j$          // The boundary surface $CB_j$ is subtracted from $A_i$ //
    end if
  end for
  if $A_i \neq \emptyset$ then
    $\mathcal{R} \leftarrow \mathcal{R} \cup A_i$                // $A_i$ is added to the remaining surfaces set $\mathcal{R}$ //
  end if
end for
```

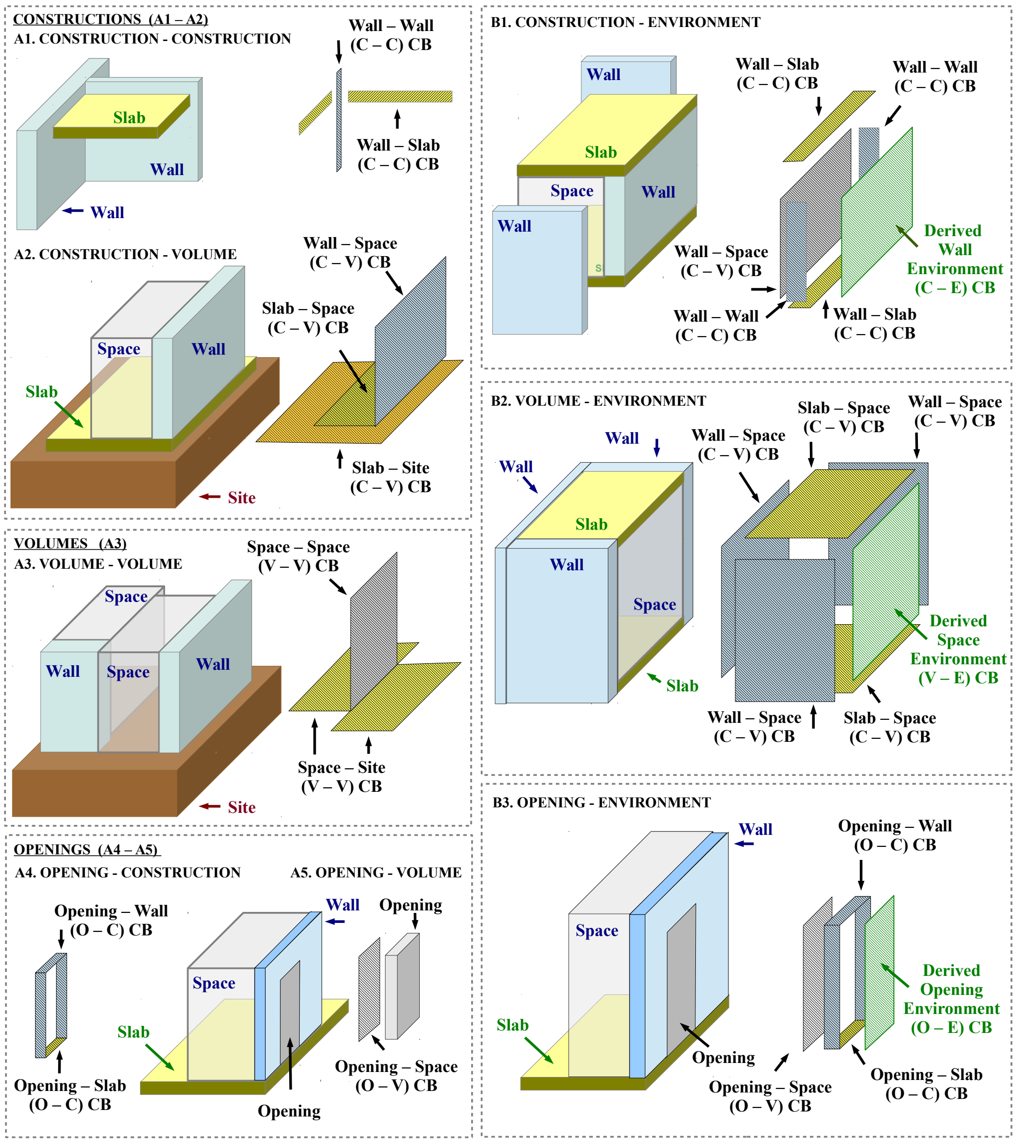


Figure 8: Primary types of common boundary surfaces referring to building constructions (A cases) and Derived environment surfaces (B cases)

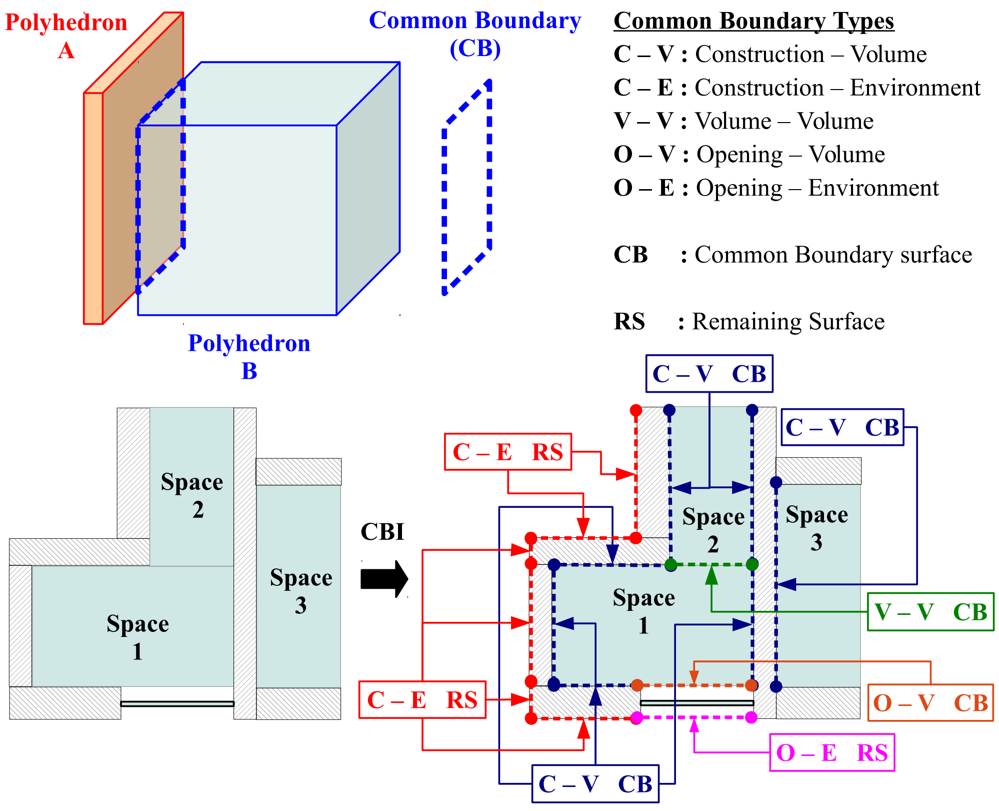


Figure 9: Geometrical illustration of CBI process on two polyhedrons (Top) – Plan view of the resulted Common Boundaries (CB) for a three space building (Bottom)

Boundary Intersection Projection stage - BIP (stage 4)

The CB surfaces are forwarded to the *Boundary Intersection Projection (BIP)* process to generate the required geometry elements (BIP elements) of a Building Energy Performance (BEP) simulation model. These elements, are essentially *surface pairs*, called CBIP surfaces or CBIPs. The BIP process can be described by two geometrical operations: (1) the *projection* of one of the common boundaries on the plane of another; and (2) the *intersection* of the projection with the other common boundary. In all cases, the surfaces of the generated BIP elements are related to a unique building entity (wall, slab, opening, etc.) and their normal vectors point away from this building entity.

The results of CBI and BIP operations for a simple example of three spaces' floor plan, are depicted in figures 9 and 10, respectively.

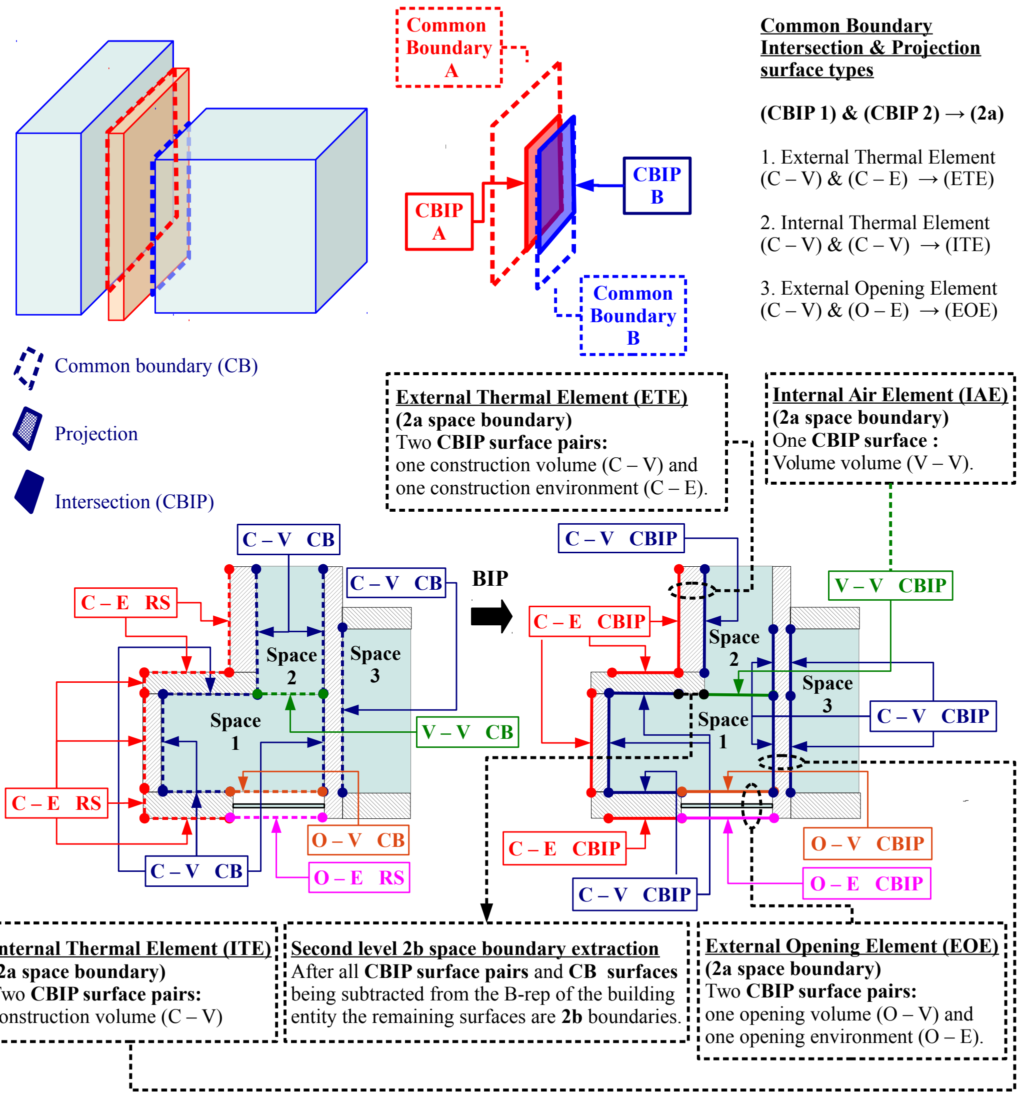

Common Boundary A
CBIP A
CBIP B
Common Boundary B
Common Boundary Intersection & Projection surface types
(CBIP 1) & (CBIP 2) → (2a)
1. External Thermal Element (C – V) & (C – E) → (ETE)
2. Internal Thermal Element (C – V) & (C – V) → (ITE)
3. External Opening Element (C – V) & (O – E) → (EOE)
Common boundary (CB)
Projection
Intersection (CBIP)
External Thermal Element (ETE) (2a space boundary) Two CBIP surface pairs: one construction volume (C – V) and one construction environment (C – E).
Internal Air Element (IAE) (2a space boundary) One CBIP surface : Volume volume (V – V).
C – V CB
C – E RS
Space 2
Space 3
Space 1
V – V CB
O – V CB
O – E RS
BIP
C – V CBIP
C – E CBIP
V – V CBIP
O – V CBIP
O – E CBIP
Internal Thermal Element (ITE) (2a space boundary) Two CBIP surface pairs: construction volume (C – V)
Second level 2b space boundary extraction After all CBIP surface pairs and CB surfaces being subtracted from the B-rep of the building entity the remaining surfaces are 2b boundaries.
External Opening Element (EOE) (2a space boundary) Two CBIP surface pairs: one opening volume (O – V) and one opening environment (O – E).

Figure 10: Geometrical illustration of BIP process on two polyhedrons (Top) – Illustration of: C-V and O-V CBIP surface pairs, V-V Common Boundaries (CB) and extracted 2b space boundary surface types, for a building space (Bottom)

The projections of BIP process are applied to four types of CBs, derived from the CBI stage: C-V (Construction - Volume), C-E (Construction - Environment), O-V (Opening - Volume), O-E (Opening - Environment). In case a C-V, C-E, O-V or O-E common boundary is not projected into a C-V, C-E, O-V or O-E common boundary, it remains as a CB and is associated with a specific building entity (wall, slab, opening, etc.). Such a case is depicted in figure 10 for a V-V CB surface of the examined space 1, which is also common boundary surface of space 2. Other possible CB surfaces not generating CBIP surface pairs are the common boundary surfaces of the C-C type.

Finally, a maximum distance criterion is used where a threshold value is used as an additional parameter, in order to exclude the surface pairs, obtained from the BIP process, whose surfaces are far apart. Additionally, an outward normal criterion is used, where only the BIP elements, whose surface pairs have normal vectors pointing away from each other, are retained. As a general rule, the normal vector of a CB surface, the direction of which is preserved during the BIP process, always starts from the first element of the CB surface and points towards the second element of the CB surface.

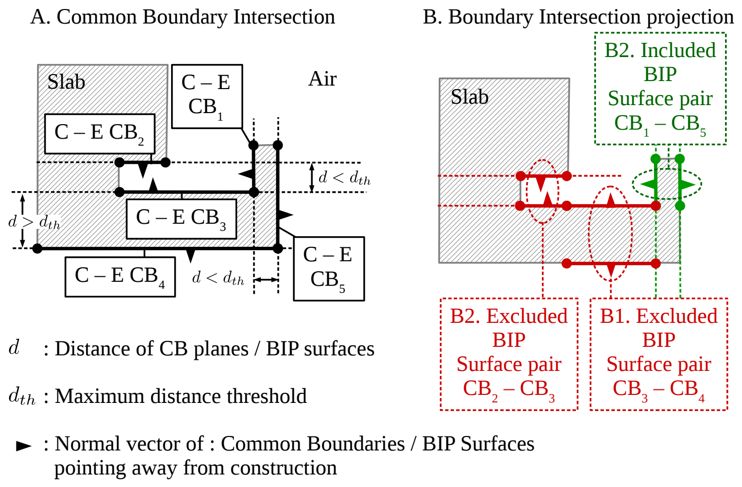

A. Common Boundary Intersection
B. Boundary Intersection projection
Slab
C – E CB1
Air
C – E CB2
d < dth
d > dth
C – E CB3
C – E CB4
d < dth
C – E CB5
B2. Included BIP Surface pair CB1 – CB5
Slab
B2. Excluded BIP Surface pair CB2 – CB3
B1. Excluded BIP Surface pair CB3 – CB4
d : Distance of CB planes / BIP surfaces
dth : Maximum distance threshold
► : Normal vector of : Common Boundaries / BIP Surfaces pointing away from construction

Figure 11: Examples of three C-E CB surface pairs referring to a single building slab (A), generating three BIP elements (B). Two of the obtained BIP elements are excluded (one violating the maximum distance criterion and one violating the outward normal criterion) and one is included

An example where three C-E CB surface pairs related to a single building slab generate three BIP elements, is displayed in figure 11. In this example, two of the obtained BIP elements obey the maximum distance threshold criterion (figure 11 part B case B2). The remaining BIP element violates the maximum distance criterion (figure 11, part B, case B1) and therefore is not included. Additionally, out of the two BIP elements which obey the maximum distance criterion, one BIP element is included since the normal vectors of its surfaces point away from each other (obeying the outward normal criterion), and one is excluded since the vectors of its surfaces pointing towards each other.

Second-level boundaries of type 2a

After the completion of BIP process on all building elements of interest, the CBIP surface pairs are obtained from the respective BIP elements. These CBIP surface pairs are related to the 2nd-level space boundaries of type 2a (as defined in [5]), and are associated to the following eight types of simulation model elements:

(1) *External Thermal Elements (ETE).*

External thermal elements are obtained by applying BIP on a Construction - Volume (C-V) / Construction - Environement (C-E) surface pair referring to the same construction entity. A common example of such element is an external wall illustrated in case A of figure 12.

(2) *Internal Thermal Elements (ITE).*

Internal thermal elements are extracted using BIP on two Construction - Volume (C-V) CB surfaces, which refer to the same construction entity. Examples include internal wall's space boundary surfaces, as displayed in case B of figure 12 and slab - space boundary pairs.

(3) *External Shading Elements (ESE).* These elements are obtained by applying BIP on Construction - Environment (C-E) CB surfaces, which refer to the same construction entity (see figure 12 case C).

(4) *Internal Shading Elements (ISE).*

Internal Shading Elements refer to construction building entities which cause shading effects inside building spaces. They are obtained by applying the BIP process on two Construction - Volume (C-V) CB surfaces, referring to the same construction and volume entities. Examples include recesses of building spaces caused by internal walls or slabs (see case D of figure 12).

(5) *External opening elements (EOE).*

External Opening Elements refer to surface pairs of building entities, that allow airflow between the environment and the building spaces. EOE are obtained by applying the BIP process on an Opening - Volume (O-V) CB surface and its Opening - Environment (O-E) CB surface counter part (see case E of figure 12).

(6) *Internal Opening Element (IOE).*

Similarly to the external opening elements, internal opening elements are represented by surface pairs of building entities, that allow airflow among building spaces. IOE are obtained using BIP on an Opening - Volume (O-V) CB surface pairs referring to the same opening element (see case F figure 12).

(7) *External Air Element (EAE).*

External Air Elements are obtained directly from V-E CB surfaces
without requiring BIP processing.

(8) *Internal Air element (IAE).*

Internal Air Elements are obtained directly from V-V CB surfaces
without requiring BIP processing.

Second-level boundaries of type 2b and 2c

Apart from the second-level boundaries of type 2a, those of type 2b, 2c and 2d [5], are also extracted. These special cases of 2nd-level space boundaries can be ignored or be entered as adiabatic surfaces in a thermal simulation model. The extraction process of 2b, 2c and 2d space boundary types is similar to the RSE process, performed by Algorithm 5. Here, the polyhedron under consideration $\mathbf{A}$, is assigned to the B-rep of a building space, and not to a construction,

Additionally, the set $\mathcal{CB}$, contains all the associated obtained CBIP surface pairs or CB surfaces (C-V, O-V CBIPs or V-E, V-V CBs), related to this space (volume) which is represented by the polyhedron $\mathbf{A}$. After executing a process similar to 5 the resulting set $\mathcal{R}$, if its is not empty, will contain the second-level space boundaries of type 2b and 2c, associated with the space volume under consideration.

An example of a 2b space boundary extraction, after all the CBIP and CB surfaces referring to a single space, are collected, is illustrated in the bottom part of figure 10.

Connectivity information

All of the previous types of elements are associated with certain connectivity information which is also required in thermal simulations. CBIP provides this information in the form of matrices of different number of entries according to Table 1.

Table 1: Connectivity information of thermal elements.
Ci. = Construction index, Isi = Internal space index, Ei = Environment index

Element	**Connectivity information**
ETE	(Ci) / (Isi) / (Ei [1])
ITE	(Ci) / (1st Isi) / (2nd Isi)
ESE	n/a
ISE	(Isi)
EOE	(Ci) / (Isi) / (Ei [1])
IOE	(Ci) / (1st Isi) / (2nd Isi)
EAE	(Isi) / (Ei [1])
IAE	(1st Isi) / (2nd Isi)

External and internal air elements have the same connectivity information with the respective external and internal thermal and opening elements, without any construction associated to them.

IFC data refinement

After the CBIP's geometric operations, the IFC data can be updated using the results of the algorithm. More precisely, the surface pairs defined by the CBIP output elements described earlier, can populate the *IfcRelSpaceBoundary2ndLevel* classes, as illustrated in the example of figure 13. The geometry of each boundary surface (surface polygon) is defined in the *ConnectionGeometry* item. The boundary's location, with respect to other building entities, is indicated by the *InternalOrExternal* item, which can potentially receive the following values:

- *INTERNAL*, if the boundary surface is attached to an internal building space (Boundaries #102, #103 and #105 in figure 13);
- *EXTERNAL*, if the boundary surface is attached to the outside air environment (Boundaries #101, #104 and #106 in figure 13);
- *EXTERNAL_EARTH*, if the boundary surface is attached to ground;
- *EXTERNAL_WATER*, if the boundary surface is attached to water;
- *EXTERNAL_FIRE*, if the boundary surface is attached to another building; and
- *NOTDEFINED*, if none of the previous cases holds.

[1]The environment index obtains two values: -1 if the environment entity refers to the building site and 0 if the environment entity refers to the outside air.

If the boundary surface refers to a building construction (wall, slab , etc.), the item *PhysicalOrVirtual* receives the *PHYSICAL* value (boundaries #101, #102, #103, #104 in figure 13); if it refers to a surface separating building spaces, the PhysicalOrVirtual receives the *VIRTUAL* value (boundaries #105 and #106 in figure 13); otherwise, PhysicalOrVirtual becomes *NOTDEFINED*.

CBIP's surface pairs are defined by the attribute *CorrespondingBoundary*. For example, the external wall of figure 13 contains a thermal element, defined by two boundary surfaces (#101, #102), which forms a pair indicated by the CorrespondingBoundary attribute (the Corresponding boundary of #101 is #102 and vice versa).

If a boundary surface contains openings (doors, windows, etc.), these openings are indicated by the *InnerBoundary* attribute of the boundary surface, which contains the boundary surface pairs of these openings. In figure 13 for example, the boundary surface #102 contains an opening indicated by the InnerBoundary #103. In the same manner, for the inner space boundaries, the space boundaries they belong to are indicated by the attribute *ParentBoundary* (as indicated in figure 13, inner boundaries #104 and #103 have boundaries #102 and #101 as parent boundaries, respectively).

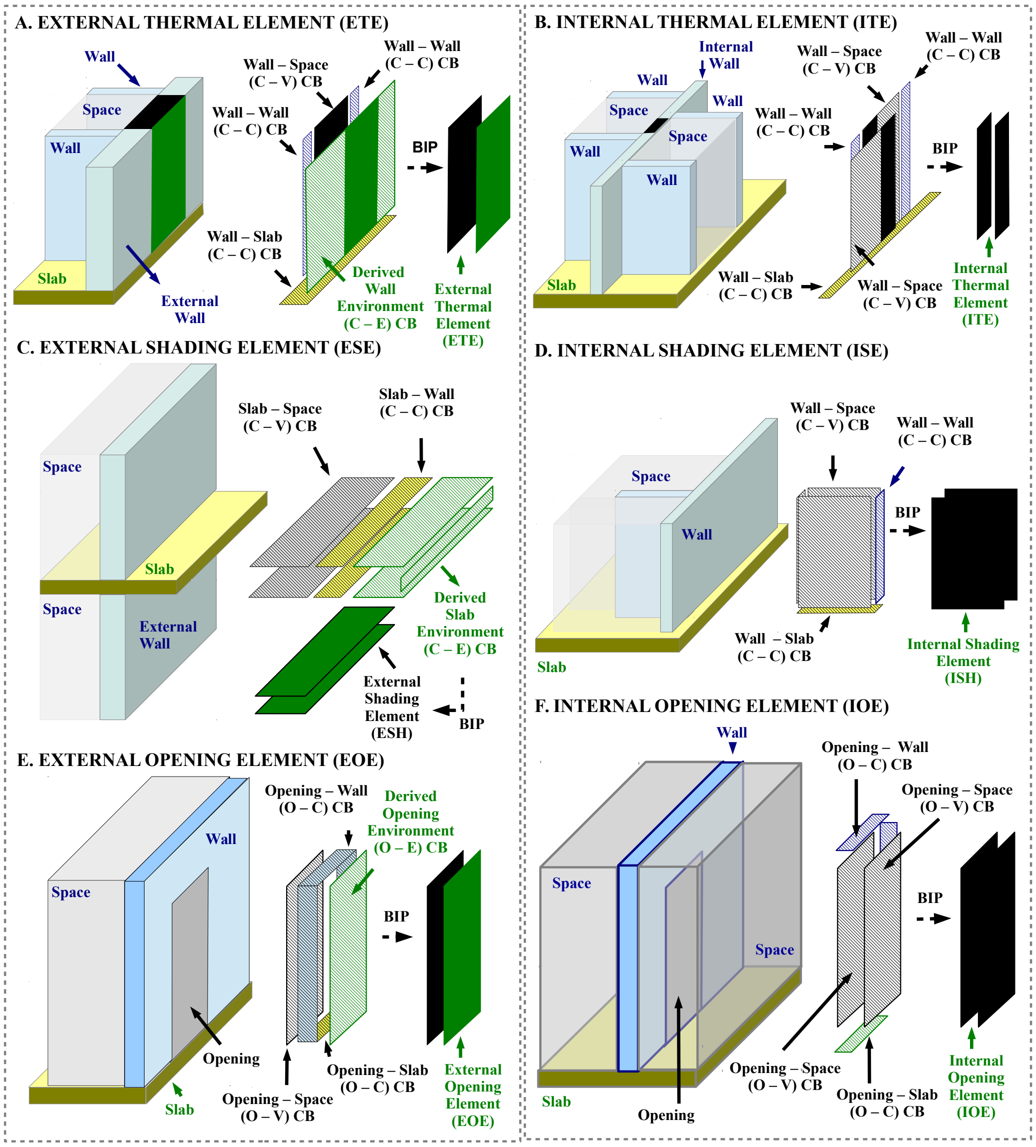

A. EXTERNAL THERMAL ELEMENT (ETE)
Wall
Space
Wall
Slab
External Wall
Wall – Space (C – V) CB
Wall – Wall (C – C) CB
Wall – Wall (C – C) CB
Wall – Slab (C – C) CB
Derived Wall Environment (C – E) CB
BIP
External Thermal Element (ETE)
B. INTERNAL THERMAL ELEMENT (ITE)
Internal Wall
Wall
Space
Wall
Space
Wall
Wall
Slab
Wall – Space (C – V) CB
Wall – Wall (C – C) CB
Wall – Wall (C – C) CB
Wall – Slab (C – C) CB
Wall – Space (C – V) CB
BIP
Internal Thermal Element (ITE)
C. EXTERNAL SHADING ELEMENT (ESE)
Space
Slab
Space
External Wall
Slab – Space (C – V) CB
Slab – Wall (C – C) CB
Derived Slab Environment (C – E) CB
External Shading Element (ESH)
BIP
D. INTERNAL SHADING ELEMENT (ISE)
Space
Wall
Slab
Wall – Space (C – V) CB
Wall – Wall (C – C) CB
Wall – Slab (C – C) CB
BIP
Internal Shading Element (ISH)
E. EXTERNAL OPENING ELEMENT (EOE)
Space
Wall
Opening
Slab
Opening – Wall (O – C) CB
Derived Opening Environment (O – E) CB
BIP
Opening – Space (O – V) CB
Opening – Slab (O – C) CB
External Opening Element (EOE)
F. INTERNAL OPENING ELEMENT (IOE)
Wall
Space
Space
Slab
Opening
Opening – Wall (O – C) CB
Opening – Space (O – V) CB
BIP
Opening – Space (O – V) CB
Opening – Slab (O – C) CB
Internal Opening Element (IOE)

Figure 12: Simulation model element examples

.

If the boundary surface refers to an internal boundary attached to a specific building space, this space is indicated by the *RelatingBuildingSpace* attribute which points to the respective IfcSpace class. For instance, in figure 13, boundaries #102, #103 and #105 indicate space #1 as their internal space.

Finally, the building element in which the boundary surface corresponds to, is indicated by the *RelatedBuildingElement*attribute. If the boundary surface is a virtual boundary, such an attribute does not exist. In figure 13, the boundaries #101 #102 refer to an external wall and the boundaries #103 and #104 refer to an external window.

Table 2 summarizes the relation between the surface pairs, obtained by CBIP, and the IFC space boundary surface types. For example, an external thermal element consists of two PHYSICAL space boundary surfaces; the first is INTERNAL facing an internal building space and the second is EXTERNAL facing the outside air environment.

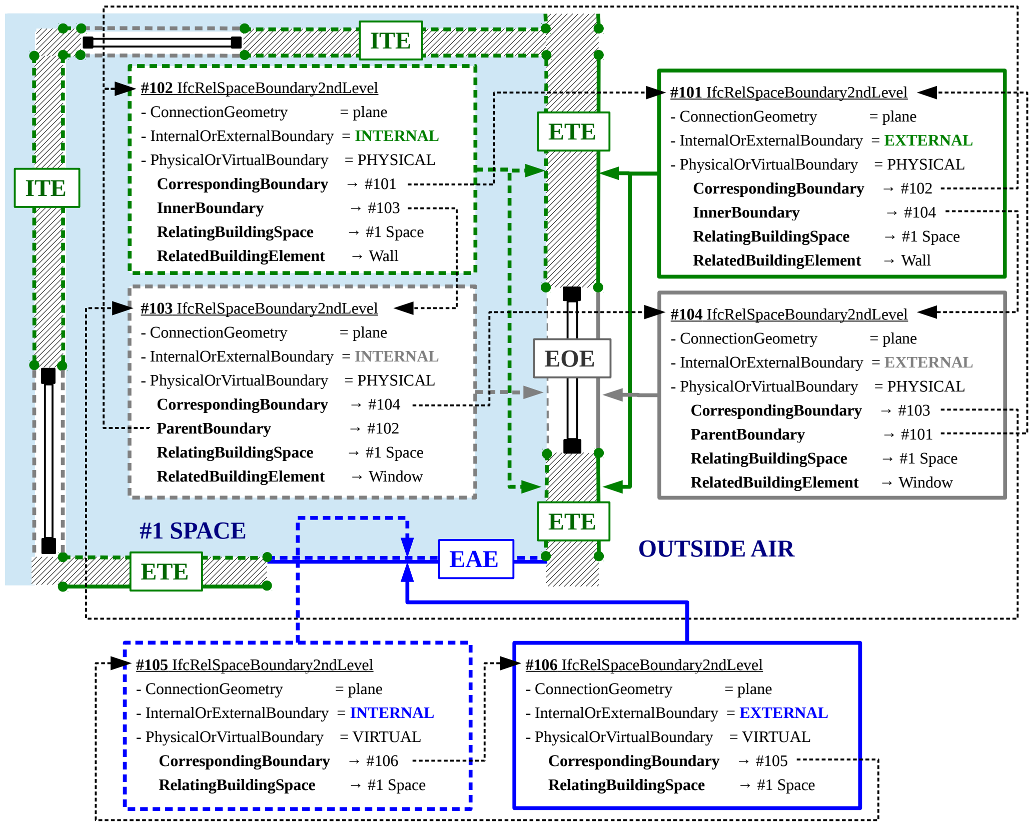

ITE
ITE
ETE
ETE
ETE
EOE
EAE
#1 SPACE
OUTSIDE AIR
#102 IfcRelSpaceBoundary2ndLevel
- ConnectionGeometry = plane
- InternalOrExternalBoundary = INTERNAL
- PhysicalOrVirtualBoundary = PHYSICAL
CorrespondingBoundary → #101
InnerBoundary → #103
RelatingBuildingSpace → #1 Space
RelatedBuildingElement → Wall
#103 IfcRelSpaceBoundary2ndLevel
- ConnectionGeometry = plane
- InternalOrExternalBoundary = INTERNAL
- PhysicalOrVirtualBoundary = PHYSICAL
CorrespondingBoundary → #104
ParentBoundary → #102
RelatingBuildingSpace → #1 Space
RelatedBuildingElement → Window
#101 IfcRelSpaceBoundary2ndLevel
- ConnectionGeometry = plane
- InternalOrExternalBoundary = EXTERNAL
- PhysicalOrVirtualBoundary = PHYSICAL
CorrespondingBoundary → #102
InnerBoundary → #104
RelatingBuildingSpace → #1 Space
RelatedBuildingElement → Wall
#104 IfcRelSpaceBoundary2ndLevel
- ConnectionGeometry = plane
- InternalOrExternalBoundary = EXTERNAL
- PhysicalOrVirtualBoundary = PHYSICAL
CorrespondingBoundary → #103
ParentBoundary → #101
RelatingBuildingSpace → #1 Space
RelatedBuildingElement → Window
#105 IfcRelSpaceBoundary2ndLevel
- ConnectionGeometry = plane
- InternalOrExternalBoundary = INTERNAL
- PhysicalOrVirtualBoundary = VIRTUAL
CorrespondingBoundary → #106
RelatingBuildingSpace → #1 Space
#106 IfcRelSpaceBoundary2ndLevel
- ConnectionGeometry = plane
- InternalOrExternalBoundary = EXTERNAL
- PhysicalOrVirtualBoundary = VIRTUAL
CorrespondingBoundary → #105
RelatingBuildingSpace → #1 Space

Figure 13: Examples of IFC4 classes of second-level space boundaries and their related building entities and CBIP output elements

.

Table 2: CBIP output elements and respective IFC space boundary types.
PHY = PHYSICAL, VIR. = VIRTUAL, INT = INTERNAL, EXT = EXTERNAL

Sim. Model Element	**1st boundary surface**	**2nd boundary surface**
ETE	PHY / INT	PHY / EXT
ITE	PHY / INT	PHY / INT
ESE	PHY / EXT	PHY / EXT
ISE	PHY / INT	PHY / INT
EOE	PHY / INT	PHY / EXT
IOE	PHY / INT	PHY / INT
EAE	VIR / INT	VIR / EXT
IAE	VIR / INT	VIR / INT

Design requirements

In order to ensure the correct execution of CBIP algorithm certain design requirements must be satisfied, which are described in the following subsections.

Building site and spaces

The building entities which are associated with the operation of CBIP must contain at least one element related to the building site and at least one element related to a building space. Such requirements are not met by the Design Transfer View 1.0 [33] model view definition.

The building site acts as a reference level in thermal simulations attaining the ground temperature, which is considerably different than the air or building interior temperatures. Consequently, its presence and relative location to other building elements is of paramount importance. On the other hand building spaces are associated with simulation output values (temperature, humidity, etc.) and therefore the presence and the geometrical definition of at least one building space is prerequisite.

Curved solids

The geometric information of any curved building entity must be exported in the IFC file considering a polyhedral approximation of the entity. This approximation must have its boundary surfaces oriented according to the right hand outward normal rule, as explained in Section 2.1. Such a requirement can be set by the exporting software.

Design recommendations

Apart from the previous mandatory site and building space requirements, there are some additional design recommendations, compliance of which guarantees accuracy of CBIP results. These recommendations are related to certain scenarios and are described in the following subsections.

Nonzero volume intersections

A nonzero volume intersection occurs when two or more building entities (wall, slab, space, etc) share a common nonzero volume, meaning that their solid geometric representations are intersected. Such cases can be identified using a model checking software such as Solibri [28]. These cases do not impede the execution of CBIP, but affect the accuracy of its results. They can be corrected manually or automatically by using the algorithms of [27]. An example of a nonzero volume intersection between a wall and a slab is displayed in the images of Case A1 (inaccurate) and Case A2 (accurate) in figure 14.

Space-environment surfaces

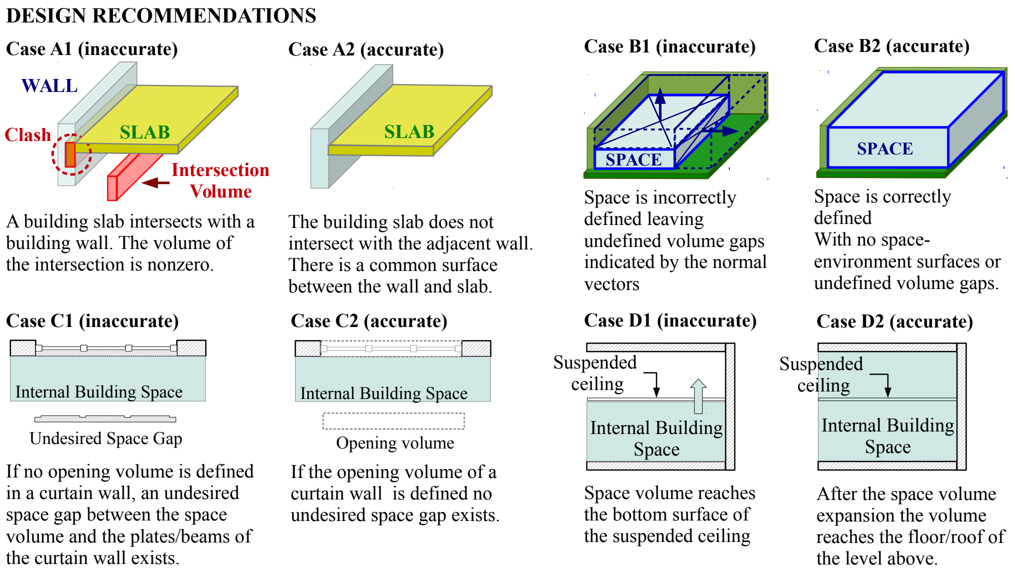
The accuracy of CBIP results is also affected by the presence of space-environment surfaces associated with internal surfaces. Space environment surfaces are derived surfaces of V-E type (see section 4.3.2), which define areas where a building space is not attached to any other building entity. These surfaces occur when an internal building space is not defined correctly, leaving small undefined space gaps between the space and surrounding building entities. In such cases the building spaces should be redefined correctly by redesign, or corrected using a space correction algorithm [27]. Examples of space-environment surfaces related to an incorrect space definition is displayed in the images of Case B1 (inaccurate) and Case B2 (accurate) of figure 14.

Curtain walls

If a curtain wall is present, it should always be contained inside an opening volume — a volume with surface area equal to the surface area of the curtain wall and thickness equal to the thickness of the wall it is attached to. This requirement is displayed graphically in the images of Case C1 (inaccurate) and Case C2 (accurate) of figure 14.

Suspended ceilings

If a suspended ceiling is present, the space volume beneath should extend to the floor (or the roof if the space is in the last level) above it. This requirement is displayed graphically by the images of Case D1 (inaccurate) and Case D2 (accurate) of figure 14. Otherwise, an additional space volume should be defined between the suspended ceiling and the floor, or roof above it.



A building slab intersects with a building wall. The volume of the intersection is nonzero.

The building slab does not intersect with the adjacent wall. There is a common surface between the wall and slab.

Space is incorrectly defined leaving undefined volume gaps indicated by the normal vectors

Space is correctly defined With no space-environment surfaces or undefined volume gaps.

If no opening volume is defined in a curtain wall, an undesired space gap between the space volume and the plates/beams of the curtain wall exists.

If the opening volume of a curtain wall is defined no undesired space gap exists.

Space volume reaches the bottom surface of the suspended ceiling

After the space volume expansion the volume reaches the floor/roof of the level above.

Figure 14: Design requirements for curtain walls (Left) and suspended ceilings (Right).

Orientation of boundary surfaces

To ensure accuracy in the results of CBIP, all of the involved building boundary representations should be complete (without missing surfaces) and their boundary surface polygons should have a right-hand outward normal orientation, as described in Section 2.1. However, not all IFC exporting programs conform to such requirements. Therefore, an outward surface normal vector check of the involved polyhedrons and corrections were necessary, are required.

Comparison with other techniques

Several other techniques have been proposed for second-level space boundary generation which take as input the three dimensional geometry contained in IFC files and generate the second-level space boundary surface pairs using building graphs, as opposed to CBIP which is a graph-less method. In [22] graphs are constructed using the faces of building B-reps as graph vertexes and the connecting edges of the B-rep faces as graph edges. In [24] graphs are created using as vertexes the space construction common boundary surface and the construction parts between them and as edges the possible thermal flow paths. Finally in [23], graphs are used in order to connect surface polygons of constructions (nodes) which ”view” one another using the definition and calculations of surface view factors.

All of the above methods do not require the geometrical definition of buildings’ conditioned space air volumes and attempt to invoke such information based on the B-reps of the surrounding constructions (walls, slabs, ...) and their connectivity. In this sense, these methods are particularly useful in cases the conditioned building space volume data are missing. In such cases CBIP cannot be applied, since it requires the geometrical definition of the conditioned building space volumes. However, in the other methods there is no reference on the calculation of external shading surfaces (shading elements) and virtual space partitions (air elements), and surfaces attached to building site (site boundaries). In this sense, CBIP can provide the space boundary surface pairs arising from virtual space partitioning, and building-site adjacency since it uses the geometrical definitions of the volumes of the spaces of the building and its site. Additionally CBIP can provide external shading surface computation which is useful for solar gain calculation routines of building energy performance simulation programs.

Demonstration example

CBIP was applied successfully in several building reference cases including the ones described in [34]. However, in order to include all possible geometrical complexities appearing in real buildings, the algorithm is demonstrated here on the Technical Services building of Technical University of Crete pictured in the upper part of figure 15. This building has two floors and features highly complex geometry containing both convex and non-convex spaces.

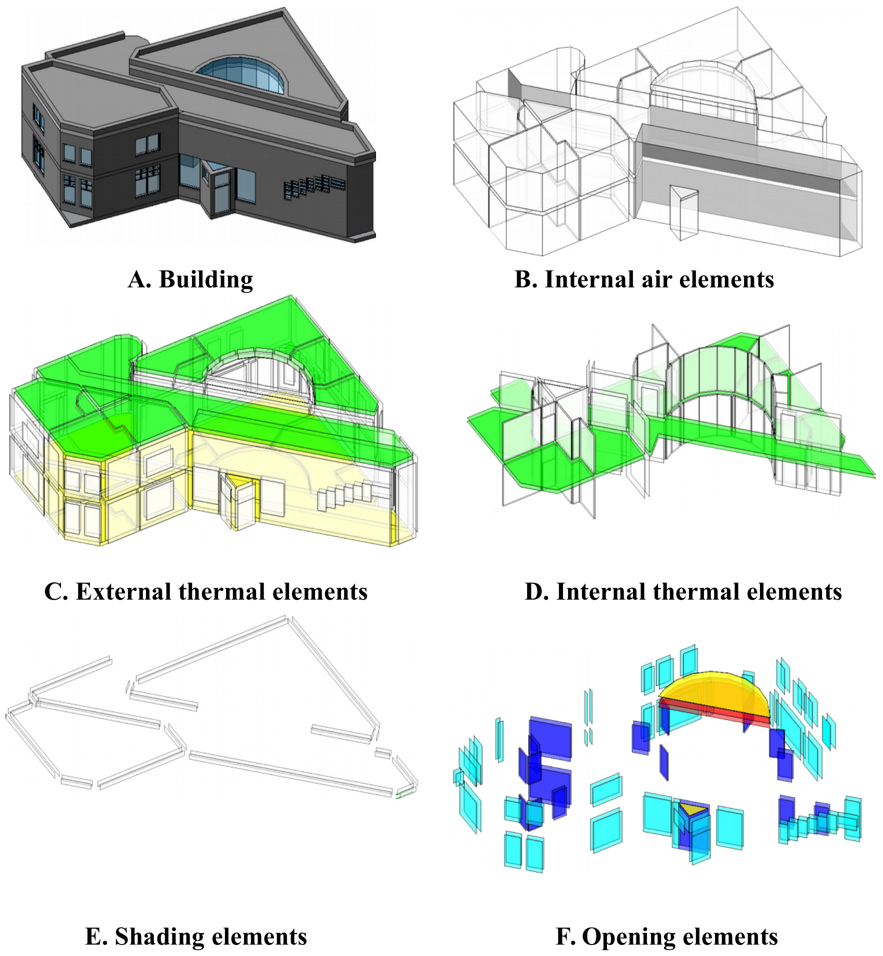


Figure 15: Demonstration building: Technical services building in Technical University of Crete (top-left) and results of CBIP on the technical services building of the Technical University of Crete

.

As figure 15 demonstrates, CBIP correctly partitions the building walls and slabs according to the topology of the building spaces. CBIP correctly identifies: the external and internal thermal elements (figure 15, parts C and D) the shading elements (figure 15, part E), the inner and outer opening elements (figure 15, part F), as well as the internal air elements (figure 15, part B) of this demonstration building. The total number of the identified elements as well as their display colors are presented in Table 3. Based on these elements, the total identified 2nd-level space boundary surfaces are 514, 122 of which are inner boundary surfaces referring to openings. During the BIP process, a maximum thickness threshold value of 1.2 meters, was used.

Table 3: Simulation model elements of demonstration building (Total number / Display color)

Element	No./color (wall)	No./color (slabs)
ETE	65 / white	14 / green - 15 / yellow
ITE	79 / white	13 / green
ESE	18 / white	1 / green
ISE	0	0
EOE	41 / cyan	2 / yellow
IOE	17 / blue	1 / red
EAE	0	0
IAE	10 / grey	0

Conclusions

It has been demonstrated that the proposed CBIP process generates the geometry of a building's 2nd-level space boundaries, which are required for building energy simulations, based on geometric information contained in IFC data files. In a nutshell, CBIP renders the process of energy simulation model generation, directly from IFC data semi-automatic. This fact, combined with the updatability and interoperability advantages of the IFC format, facilitates the data exchange between the BIM and energy simulation programs and enables a continuous building performance monitoring.

In order to perform these operations CBIP uses the polyhedral geometric representations of the building entities. As its name reveals CBIP's operation is based on two main subprocesses: the common boundary intersection (CBI) subprocess and the boundary intersection projection (BIP).

CBIP was applied on a highly complex building and the results demonstrated the ability of the algorithm in handling non-convex geometries generating all the possible types of simulation model elements including: thermal, opening, shading and air elements (virtual space boundaries). The second-level space boundaries were identified and their space connectivity information was obtained accurately. In conclusion, CBIP automates the transformation process of IFC geometric data, to all the geometric data required for the creation of energy simulation models.

Acknowledgments

The research leading to these results has been partially funded by the European Commission FP7-ICT-2011-6, ICT Systems for Energy Efficiency under contract #288409 (BaaS) and by the European Commission H2020-EeB5-2015 project Optimised Energy Efficient Design Platform for Refurbishment at District Level under contract #680676 (OptEEmAL).

[1] J. Clarke, Energy simulation in building design, Taylor & Francis, ISBN: 978-0-750-65082-3, 2012.

[2] K. Pratt, N. L. Jones, L. Schumann, D. Bosworth, A. Heumann, Automated translation of architectural models for energy simulation, in: Proc. of Symposium on Simulation for Architecture and Urban Design, 6, Orlando,USA, 2012.

[3] G. Kontes, G. Giannakis, E. B. Kosmatopoulos, D. Rovas, Adaptive-fine tuning of building energy management systems using co-simulation, in: Proc. of IEEE International Conference on Control Applications, Zagreb, Croatia, 1664–1669, DOI: 10.1109/CCA.2012.6402707, 2012.

[4] G. D. Kontes, C. Valmaseda, G. I. Giannakis, K. I. Katsigarakis, D. V. Rovas, Intelligent BEMS design using detailed thermal simulation models and surrogate-based stochastic optimization, Journal of Process Control 24 (6) (2014) 846–855, DOI: 10.1016/j.jprocont.2014.04.003.

[5] V. Bazjanac, Space boundary requirements for modeling of building geometry for energy and other performance simulation, in: Proc. CIB W78: 27th International Conference, Cairo, Egypt, 2010.

[6] C. Eastman, P. Teicholz, R. Sacks, K. Liston, BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors, Wiley, ISBN: 978-0-470-54137-1, 2011.

[7] T. Liebich, IFC4-the new buildingSMART standard, in: IC meeting, bSI Publications, Helsinki, Finland, 2013.

[8] gbXML, The Green Building XML schema, <http://www.gbXML.org>, 2011.

[9] AutoDesk, Revit^{TM}, <http://www.autodesk.com/products/autodesk-revit-family/overview>, 2014.

[10] GraphiSoft, ArchiCAD 17^{TM}, <http://www.graphisoft.com/archicad/>, 2013.

[11] V. Bazjanac, Implementation of semi-automated energy performance simulation: building geometry, in: CIB W, vol. 78, 2009.

[12] U.S. Dept. of Energy, EnergyPlus, <http://apps1.eere.energy.gov/buildings/energyplus/>, 2014.

[13] J. O'Donnell, T. Maile, C. Rose, N. Mrazovi, E. Morrissey, C. Regnier, K. Parrish, B. V., Transforming BIM to BEM: Generation of Building Geometry for the NASA Ames Sustainability Base BIM, Tech. Rep., Simulation Research Group, 2013.

[14] M. Jokela, A. Keinänen, H. Lahtela, K. Lassila, O. Granlund, Integrated building simulation tool RIUSKA, in: Proc. of Building Simulation IBPSA conf., Prague, Czech Republic, 1997.

[15] J. J. . A. Hirsch, DOE 2.1, <http://www.doe2.com>, 2014.

[16] A. Karola, H. Lahtela, R. Hänninen, R. Hitchcock, Q. Chen, S. Dajka, K. Hagström, BSPro COM-Server interoperability between software tools using industrial foundation classes, Energy and Buildings 34 (9) (2002) 901–907, DOI: 10.1016/S0378-7788(02)00066-X.

[17] AutoDesk, Green Building Studio$^{\text{TM}}$, <https://gbs.autodesk.com/GBS/>, 2014.

[18] T. Maile, M. Fischer, V. Bazjanac, Building energy performance simulation tools-a life-cycle and interoperable perspective, Center for Integrated Facility Engineering (CIFE) Working Paper 107 (2007) 1–49.

[19] TRNSYS, TRNSYS 17$^{\text{TM}}$, <http://www.trnsys.com/>, 2014.

[20] IES, VE-pro$^{\text{TM}}$, <http://iesve.com/software/ve-pro>, 2014.

[21] R. Hitchcock, J. Wong, Transforming IFC Architectural View BIMs for Energy Simulation : 2011, in: Proc. of Building Simulation IBPSA conf., Sydney, Australia, 1089–1095, 2011.

[22] C. van Treeck, E. Rank, Dimensional reduction of 3D building models using graph theory and its application in building energy simulation, Engineering with Computers 23 (2) (2007) 109–122, DOI: 10.1007/s00366-006-0053-7.

[23] N. L. Jones, C. J. McCrone, B. J. Walter, K. B. Pratt, D. P. Greenberg, Automated translation and thermal zoning of digital building models for energy analysis, in: Proc. of Building Simulation IBPSA conf., ISBN: 978-1-61839-790-4, 202–209, 2013.

[24] C. Rose, V. Bazjanac, An algorithm to generate space boundaries for building energy simulation, Engineering with Computers (2013) 1–10 .DOI: 10.1007/s00366-013-0347-5.

[25] G. N. Lilis, G. I. Giannakis, G. Kontes, D. V. Rovas, Semi-automatic thermal simulation model generation from IFC data, in: Proc. of European Conference on Product and Process Modelling, Vienna, Austria, 503–510, DOI: 10.1201/b17396-83, 2014.

[26] G. N. Lilis, K. Sklivaniotis, G. Giannakis, D. Rovas, SRC: A systemic approach to building thermal simulation, in: Proc. of Building Simulation IBPSA conf., 3192–3199, 2013.

[27] G. N. Lilis, G. I. Giannakis, D. V. Rovas, Detection and semi-automatic correction of geometric inaccuracies in IFC files, in: Proc. of Building Simulation IBPSA conf., Hyderabad, India, 2182–2189, 2015.

[28] L. Khemlani, Solibri Model Checker$^{\text{TM}}$, CADENCE-AUSTIN (2002) 32–34.

[29] A. Borrmann, S. Schraufstetter, E. Rank, Implementing metric operators of a spatial query language for 3D building models: octree and B-Rep approaches, Journal of Computing in Civil Engineering 23 (1) (2009) 34–46, DOI: 10.1061/(ASCE)0887-3801(2009)23:1(34).

[30] D. J. Jackson, Boundary representation modelling with local tolerances, in: Proc. of the third ACM symposium on Solid modeling and applications, ACM, 247–254, DOI: 10.1145/218013.218067, 1995.

[31] W. Thibault, B. Naylor, Set operations on polyhedra using binary space partitioning trees, in: Proc. of the 14th annual conference on Computer graphics and interactive techniques, vol. 21, ACM, 153–162, DOI: 10.1145/37401.37421, 1987.

[32] B. Vatti, A generic solution to polygon clipping, Communications of the ACM 35 (7) (1992) 56–63, DOI: 10.1145/129902.129906.

[33] buildingSMART, Design Transfer View version 1.0, <http://www.buildingsmart-tech.org/mvd/IFC4Add1/DTV/1.0/html/>, 2015.

[34] M. Weise, T. Liebich, R. See, V. Bazjanac, T. Laine, B. Welle, Implementation guide: Space boundaries for energy analysis, General Services Administration (GSA) and Open Geospatial Consortium (OSC) (2011) 1–62.