To be published as "Dimyadi J, Solihin W., Amor R (2018). Using IFC to Support Enclosure Fire Dynamics Simulation. In Advanced Computing Strategies for Engineering, Springer International Publishing. doi: 10.1007/978-3-319-91638-5_19

Using IFC to Support Enclosure Fire Dynamics Simulation

Johannes Dimyadi[1[0000-0002-0004-1131]], Wawan Solihin[2[0000-0001-7538-5265]], and Robert Amor[3[0000-0002-4329-9044]]

[1, 3] University of Auckland, Auckland, New Zealand
{jdim006, trebor}@cs.auckland.ac.nz
[2] novaCITYNETS Pte. Ltd., Singapore
w.solihin@outlook.com

Abstract. One objective of the performance-based design (PBD) of fire safety in buildings is to ensure occupants have an adequate time to escape the effects of the fire unharmed. A commonly accepted threshold of the safe evacuation time is the time it takes for the fire to compromise the escape routes rendering them untenable. This is a complex computational problem that often requires simulations to solve given the geometry of the building, plausible fire scenarios, thermophysical properties of building materials and furnishing, as well as the environmental conditions.

Conventionally, the input data preparation for such simulations is time consuming and error-prone as it involves tedious manual measurement and data transcription from paper-based drawings and specification documents. The recent uptake of ISO-based building information modelling (BIM) among building designers means that such information should become more readily available. However, sharing information from such a highly complex data model as BIM with downstream applications such as fire dynamics simulations can be challenging due to the lack of a practical method for querying spatial data. This paper explores two methods of sharing BIM information with an industry standard enclosure fire dynamics simulation tool. The paper further describes potential future work in using the simulation output to help auditing a given scenario against a set of compliance criteria. A four-storey sample building model is presented in the paper to illustrate both approaches.

Keywords: BIM, IFC, FDS, Geometric representation

1 Introduction

1.1 Sharing Building Information Modelling (BIM) Data

The BIM (Building Information Modelling) technology has enabled a collaborative approach to design and construct buildings in recent years as it is progressively being adopted by building design practitioners. BIM has the potential of making available one set of common building information to share among processes and applications in a building project. One way of achieving this, in theory, is by using the ISO16739 IFC

(Industry Foundation Classes) data model [1]. IFC is a highly-structured, rich and necessarily complex data model to represent every physical and functional aspect of such a complex object as a building. Extracting the geometry and spatial data from an IFC file is a significant undertaking due to the complexity of the data structure. Although commercial tools offering such data extract functionality are available, they are proprietary in nature and offer limited query capabilities and integration options with other software interfaces. Non-proprietary tools such as the open source BIMserver [2] are also available and may be used as a common platform to share building information, but additional development work is needed to augment its basic querying capabilities. There are also software add-ons, such as SimLab IFC Importer for SketchUp [3] and IfcBlender [4] for Blender [5], but they are designed to extend the capabilities of specific 3D model authoring software applications with the IFC import and export functionality.

Recent research to address the gap has seen the development of BIMRL (BIM Rule Language) that offers an efficient IFC data querying capability that supports multiple geometric representations [6]. BIMRL can act as an independent IFC query engine to service any software interface that requires BIM data for any application.

This paper describes the use of BIMRL as a query engine to provide the required BIM data from an IFC file for mapping to the input data specification of a fire dynamics simulator. The paper also presents a comparative approach of getting BIM data using Blender 3D modelling application [5] with third-party software add-ons to achieve the same objective.

1.2 Fire Safety Compliant Design Objectives

The fire safety of occupants is one of the most important considerations when designing a building. Every legal framework in the world would stipulate some forms of requirement for a building to be designed with an adequate means of escape for occupants in the event of an emergency. An adequate means of escape from a fire in a building is commonly measured in terms of the time it takes for occupants to reach a safe point outside the building before the escape routes become compromised by the effects of the fire. Another common objective is to ensure that the structural integrity is maintained sufficiently for fire- fighting operations. This can represent a complex engineering design problem involving several fire safety science principles that deal with how fire develops, its impact on the structure, and how toxic products of combustion would spread within a building enclosure.

Building codes and regulations traditionally prescribe solutions that are based on a typical building with a limited set of parameters. These “one-size-fits-all” prescriptive requirements specify exactly how a building must be constructed so that it can be deemed to satisfy the safety objectives. Modern legal frameworks incorporate the performance-based design (PBD) approach that allows designers to offer a unique and innovative solution that would satisfy the performance objectives, which are usually qualitative in nature. The PBD approach often requires that the design solution be validated using established scientific principles to demonstrate that it is valid for the building and

its intended usage throughout its entire life-cycle. Fire engineers often resort to numerical simulations for this purpose, which requires an accurate geometry of the space and essential information such as the furnishing, fenestration, thermo-physical properties of the building materials, and environmental conditions.

1.3 Enclosure Fire Dynamics Simulations

Enclosure Fire Dynamics pertains to the study of the fire behavior and the environmental response to the fire within an enclosure where there is a limited supply of fuel and oxygen. There are three common physics based approaches to simulating enclosure fire dynamics, namely lumped parameter or "zone models", computational fluid dynamics (CFD) models based on classical turbulence modelling techniques, and large eddy simulations (LES) techniques [7]. The latter provides the most realistic description of fire phenomena developed to date.

Two commonly used software tools in the industry today for simulating enclosure fire dynamics are B-RISK, a combined probabilistic/deterministic "zone model" developed by BRANZ (Building Research Association of New Zealand) [8] and FDS (Fire Dynamics Simulator), a CFD model developed by NIST in the US [9]. These tools represent two different types of computational fire model with distinct underlying philosophies in representing fire scenarios within an enclosure. Consequently, they also have different approaches in representing the building geometry for simulation purposes. Zone models such as B-RIK treat a building as a series of rectangular rooms interconnected to each other or to the outdoor by common openings of specified dimensions. CFD models such as FDS rely on the exact placement of solid objects, openings, and an accurate geometry of the building to determine the interaction between the fire and the environment in each enclosure.

This paper will mainly focus on the transfer of information from IFC-based BIM to FDS.

1.4 FDS (Fire Dynamics Simulator)

FDS (version 6.6 at the time of writing) defines each fire scenario within a three-dimensional computational domain consisting of rectilinear meshes with each mesh divided into three-dimensional cuboid cells [9]. FDS treats solid objects within the computational domain as flow obstructions, whereas openings such as doors and windows as flow passages. FDS employs LES numerical techniques to solve large scale hydrodynamics turbulence appropriate for low speed, thermally-driven fluid flow with an emphasis on smoke and heat transport from fires. Calculations occur in each cell and the result of the calculations become a new set of input parameters to the calculations in the adjoining cells. Depending on the input specification, FDS can determine various aspects of a simulated enclosure fire incident such as the level of thermal radiation and convection, the amount of fire products generate, and the level of concentrations of certain gas species. Some of the simulation output parameters can be used in an audit to determine if the assumed fire scenarios would be acceptable against prescribed legal thresholds.

The outcome of FDS simulation can be visualised in a number of ways using the companion tool SmokeView [10], depending on the output parameters specified in the input file.

1.5 Similar Research

Similar research was conducted in 2007 to investigate the use of a purpose-built IFCSTEP Parser tool, developed at the University of Christchurch in New Zealand, to extract a subset of the IFC data model for specific applications in the fire engineering domain [11]. A software interface, IFCSTEP-FDS, was developed in a subsequent research project to map the building geometry information extracted by the IFCSTEP Parser tool to an input dataset for FDS [12]. Several single-storey building models were used in the research to illustrate the approach. As IFCSTEP was intended to be a generic IFC parser for a specific domain, some calculations were necessary in the mapping process to suit the FDS specification.

IFCSTEP Parser was a research tool that only supported the axis-aligned bounding box (AABB) geometry representation. Consequently, it is limited to parsing objects that conform to the AABB geometry. Another limitation of IfcSTEP parser is the dependency on a specific software library that no longer supports later versions of the IFC specification.

2 FDS Input and Output Data Modelling

2.1 Geometry Specification in FDS

FDS treats solid obstructions such as walls within a building as a series of orthogonal cuboids with respect to each other. Each cuboid is represented by a bounding box and specified using the coordinates of the Lower Left Bottom (LLB) and Upper Right Top (URT) corners. The object is defined using the OBST namelist group in a line of data specification between & and / delimiters, as follows:

```
&OBST XB=LLBx,URTx,LLBy,URTy,LLBz,URTz, SURF_ID="Wall" /
```

In the above solid obstruction data specification, ***LLBx*** denotes the x-coordinate of the LLB corner of the bounding box, and ***URTy*** denotes the y-coordinate of the URT corner of the bounding box, etc. The **SURF_ID** is an optional parameter that identifies which specification of boundary condition parameters is applicable to this solid obstruction. For example, a SURF namelist group with **ID="Wall"** may specify the material composition and properties of the object. Boundary conditions of solid obstructions may also be specified with surface temperature, heat-flux, and other quantities or as a fire source. Another relevant parameter on the **SURF** namelist group is **GEOMETRY ="CYLINDRICAL"** or **"SPHERICAL"**, which is useful to represent non-planar objects such as conduits or services ducts.

The computational efficiency in FDS is partly achieved using a rectilinear numerical grid system, which is geometrically satisfactory for most building elements. However,

this may represent a limitation when certain objects have geometric features that do not conform to the rectilinear grid. In such a case, the non-conforming objects must be voxelised into multiple orthogonal cuboids that conform to the grid (**Fig. 1**). For example, a wall that is non-orthogonal with respect to the axes must be transformed into a set of voxels that best approximate its original shape while conforming to the grid. In terms of calculations, the impact of this "sawtooth" profile on the boundary conditions is currently handled by removing one of the vorticity terms in the equations of the flow solver, which is triggered by a Boolean parameter input specification on the affected object. The future release of FDS is planned to employ a high-order immersed-boundary method (IBM) of calculations for better handling this phenomenon.

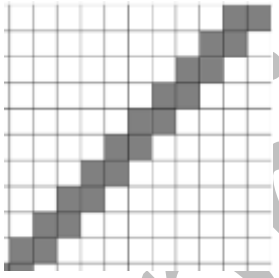

Fig. 1. Voxelised representation of a non-orthogonal object

2.2 Data Requirements for FDS

The following IFC elements are required for the purposes of FDS simulation:

- Walls (internal and external)
- Openings in walls (for doors and windows)
- Slabs
- Major structural columns or beams
- Stairs
- Major furnishing items such as desks, cabinets, or major equipment.

For solid obstructions such as walls, slabs, and major columns or beams, the following properties and attributes are also required:

- The Globally Unique Identifier (GUID) of the object
- The short and long name as well as description of the object
- Coordinates of LLB and URT corners of each object represented as a bounding box (in the World Coordinates System)
- For openings, an additional data is required, i.e. the sill height. This is particularly relevant for windows or any opening that is at certain height above the floor level.
- The materials composition of walls, their layer index, name, and thickness.

3 Querying and Extracting IFC Data

There are several considerations that influence the preferred approaches in getting the appropriate data from IFC. One important consideration is the ease of getting the right information. IFC is a standard specification designed for data exchange, which makes it very verbose and explicit. While it is an open standard that allows everyone to "see" the specification and the data, one will still need quite an in-depth understanding of the concept and the structure. This is made worse when one tries to connect two domains that deal with the same basic information but in a different view point and in different details, such as IFC and FDS. Many have tried to build the "bridge" between the two systems. In general, such system becomes difficult to manage because of the burden to deal with both systems in rather detailed knowledge.

The rise of an approach to treat IFC as a database, as evidenced in the gaining of popularity of the open source BIMserver [2], provides a little relief to this woe, but it is still inadequate. One of the difficulties is getting the right geometry data. The geometry by nature is complex, and there is no exception with IFC. As alluded to in Section 1 of this paper, for example, BIMServer can serve the IFC geometry data, but it still requires the consuming application to re-construct the data into the appropriate form needed for the application.

3.1 The BIMRL approach

The following are two main considerations behind the use of the BIMRL query language approach to share BIM data with FDS:

1. BIMRL is a simplified database representation of the IFC model. As a database, it allows flexible queries to be performed against the data. This includes a pre-check process to ensure the model has everything required to generate a good quality data needed for FDS. Being built on top of a standard RDBMS (Relational Database Management System), BIMRL queries can be performed using standard SQL statements given certain level of knowledge on its simplified schema and an understanding of the IFC data structure [13]. Using a generic tool like BIMRL releases the burden for the application to deal with a lot of details in parsing the IFC data.
2. BIMRL has a built-in concept to support multiple geometry representations [14]. This concept allows integrated queries for both building elements and their geometries. This includes not only the specification to re-construct the geometry, but the shapes of the final geometry. BIMRL generates five geometric representations for each object, i.e. AABB, OBB, Octree approximation, triangulated Polyhedron, and BREP (Boundary Representation) by a set of enriched boundary faces (**Fig. 2**). This may represent a redundancy, but also a flexibility of having different level of details available for various applications. For example, AABB data is adequate to represent orthogonal wall objects for FDS data, but the Octree cells representation is necessary to represent voxelized objects in the case of non-orthogonal walls or objects with a complex profile. All five geometry representations are pre-computed and persisted in the database for querying efficiency.

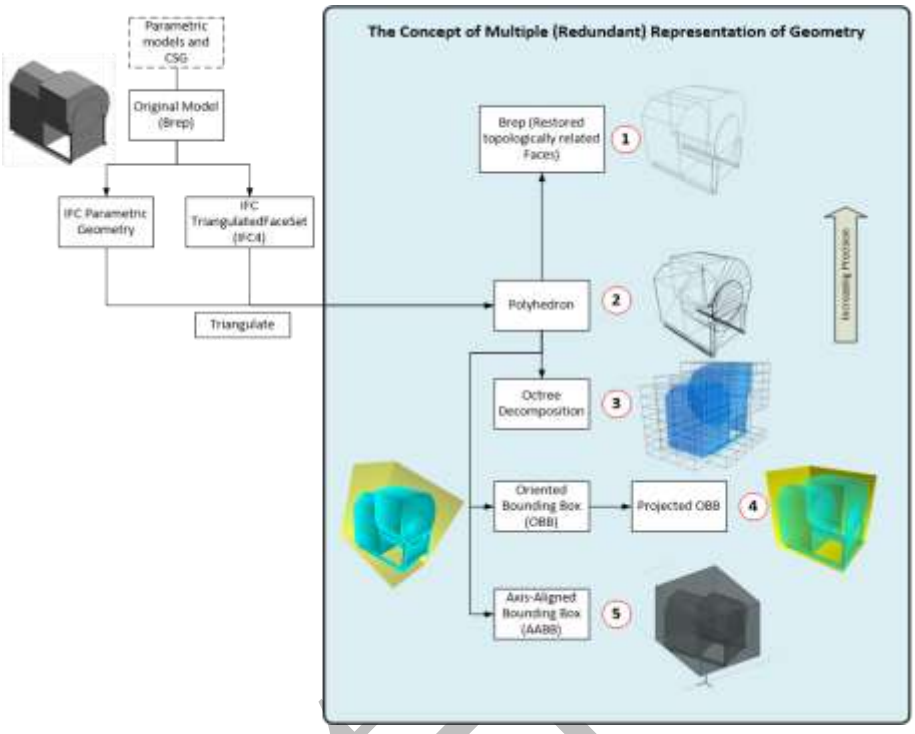

Fig. 2. BIMRL Multiple Geometry Representations [14]

Given its support for multiple geometry representations, BIMRL can extract the required data for output in a form that would minimise downstream calculations needed for the mapping to FDS. BIMRL can express the query output in JSON (Javascript Object Notation) [15], which is most suitable for data exchange using web service applications. Below is the data requirements schema (in JSON) that can be used to design the corresponding BIMRL query:

```
{
   StoreyId (string),
   StoreyName (string),
   StoreyLongName (string),
   Objects (Array of objects):
   {
      ElementId (string),
      ElementType (string),
      Name (string),
      Description (string),
      Material (array of material objects):
      [{
```

```
          LayerNo (int),
          Name (string),
          Thickness (double)
        }]
        IsOrthogonal (bool),
        CellBbox (array of LLB and URT coordinates)
        [{
           LLB:
            {x, y, z}
           URT:
            {x, y, z}
         }]
         SingleBox (an object)
        {
            Bbox (object),
            Openings (an array of objects)
             [{
                 SillHeight (double),
                 Bbox (CellBbox object)
              }]
         }
   }
}
```

BIMRL supports two types of data query, namely a standard triplet form and the uninterpreted SQL pass-through. Below is an excerpt of the BIMRL triplet query based on the data requirements schema:

```
CHECK
//Collect relevant objects, i.e. IfcWall (and
IfcWallStandardCase), IfcSlab, IfcColumn, and IfcStair
from the model into Set1
{(IfcWall, IfcSlab, IfcColumn, IfcStair) E
      Collect E.ElementId Eguid, E.ElementType Etype,
      E.Name Ename, E.Description Edesc, E.container cont
} as Set1;

//Collect all external walls (with IsExternal property
equals to TRUE) and add them into Set2
{IfcWall W Where Property(W,IsExternal)='true'
   Collect W.ElementId Eguid, Property(W,IsExternal)
   ExternalWall
} as Set2;
```

```
//Evaluate data after a Left Outer Join between Set1 and
Set2 based on the element guid (marking all objects plus
the external walls)
EVALUATE FireDataOutput(Eguid, Etype, Ename, Edesc,
   cont,"e:\\ifc2fds\\FDSOut.json") From Set1
   Left Outer Join Set2 using (Eguid);

//Print the result to the UI and to the JSON output file
ACTION print result;
```

In the above query, if the element returned an IfcWall object, then the query further checks to see if the object has a property "IsExternal" with a value of "true" indicating that it is an external wall. This is useful to identify as external walls can then be set with a transparency value in FDS to improve the visualisation of the simulation output. The extent of the computational domain can also be aligned with the outer face of the walls.

Based on the actual object geometry, BIMRL can automatically determine which geometric representation (AABB or Octree Cells) is suitable to represent the object for FDS. As explained earlier, Octree Cells would produce voxels to conform to the FDS geometry specification.

3.2 The Blender Approach

Blender is a community supported open-source generic 3D modelling tool that supports Python scripting natively for extending its basic functionality [5]. Consequently, there are many open-source add-ons available. IfcBlender is one of these add-ons, which is part of the IfcOpenShell software toolkit developed by Thomas Krijnen at the University of Eindhoven in the Netherlands [4]. IfcBlender adds the IFC geometry import functionality to Blender. Another add-on relevant for the work described in this paper is BlenderFDS [16], which allows one to compile the required data from within the Blender's modelling environment to create the input dataset for FDS.

The IFC model import for Blender is a straightforward and relatively efficient process. Once the building model is imported, the object geometry and main properties (GUID, name, and description) would then be readily available for access.

4 Generating FDS Input Data

FDS is a command line software application without any graphical user interface. Each fire scenario is defined in a single text file containing namelist groups describing different aspects of the scenario to be simulated such as the duration of the simulation, the size of the computational domain, the specification of solid obstructions, openings, and material properties, boundary conditions, and output quantities, and so on.

Several third-party pre-processors are available to assist with the creation of the FDS input. Exemplar commercial tools include Pyrosim [17], and more recently CYPECAD MEP. Exemplar open-source software add-ons and scripts include BlenderFDS [17] for

Blender, 3dsolid2fds [18] for AutoCAD, and step2fds [19]. Neither FDS nor these third-party user-interface tools currently support extracting data directly from the IFC.

4.1 The BIMRL Approach

BIMRL uses a specified data requirements schema (in JSON) as the basis for a query on a given IFC model. The query extracts the required information and outputs the result in the specified form (in JSON), which is then parsed by a JSON parser and mapped to the FDS input dataset in a single process (**Fig. 3**). For this exercise, the BIMRL query result is made available as a physical file for processing by a purpose-built FDS input data mapping interface, which incorporates a JSON parser. An alternative data exchange method could be via a web service application, which enables a remote application to make use of the query service provided by BIMRL.

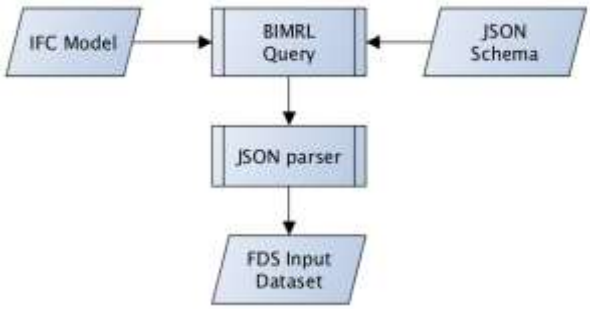

Fig. 3. IFC to FDS Data Mapping using BIMRL

The BIMRL approach enables the FDS data mapping process to be integrated, for example, with an automated compliance audit framework that can interface with FDS to support the PBD approach.

4.2 The Blender Approach

The Blender approach uses two software add-ons to achieve the same objective. Firstly, IfcBlender add-on tool is used to import the geometry of the IFC model of the sample building. Once imported, the next step is to use the BlenderFDS add-on interface to manually classify individual objects into the desired type of representation, i.e. either as AABB or as Voxels. This is a time-consuming process that is dependent on the placement and geometry of the object in relation to the intended computational grid in FDS. As each object is classified, BlenderFDS would create the resulting OBST namelist group formatted line of data for that object and save it into a collection. Eventually, all objects are classified and the lines of data in the collection can then be compiled and exported as the required FDS input dataset (**Fig. 4**).

BlenderFDS is an interface within the Blender modelling environment to guide the user in creating the FDS input data, which includes non-geometrical information such as material properties definition, computational domain setup and other parameters.

However, these information needs to be entered manually by the user as part of the FDS input data modelling.

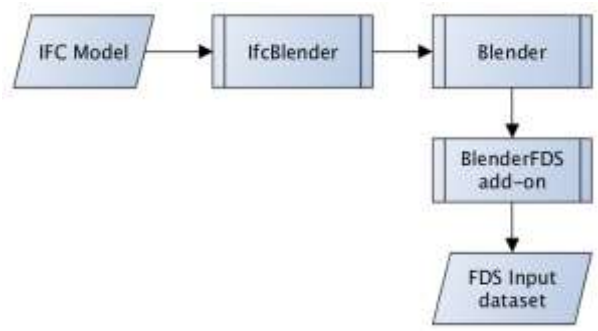

Fig. 4. IFC to FDS Data Mapping using Blender

The Blender approach effectively provides a graphical user-interface to FDS with the advantage of being able to share the BIM geometry data.

5 Worked Example

A sample four-storey building model has been selected to illustrate the approach of sharing BIM data described in this paper. This building model was originally developed by the regulatory authority in New Zealand as a test case for the performance-based compliant fire safety design verification method [20]. The actual model used for the work described in this paper was developed using ARCHICAD 20 and exported to IFC2x3.

Fig. 5 shows the exported model being viewed in the IFC viewer tool, a part of the IFC Tools Project by APSTEX. New features such as fenestrations and a non-orthogonal internal wall have been added to the model to demonstrate the challenges associated with mapping the geometry data to suit the input specification of FDS.

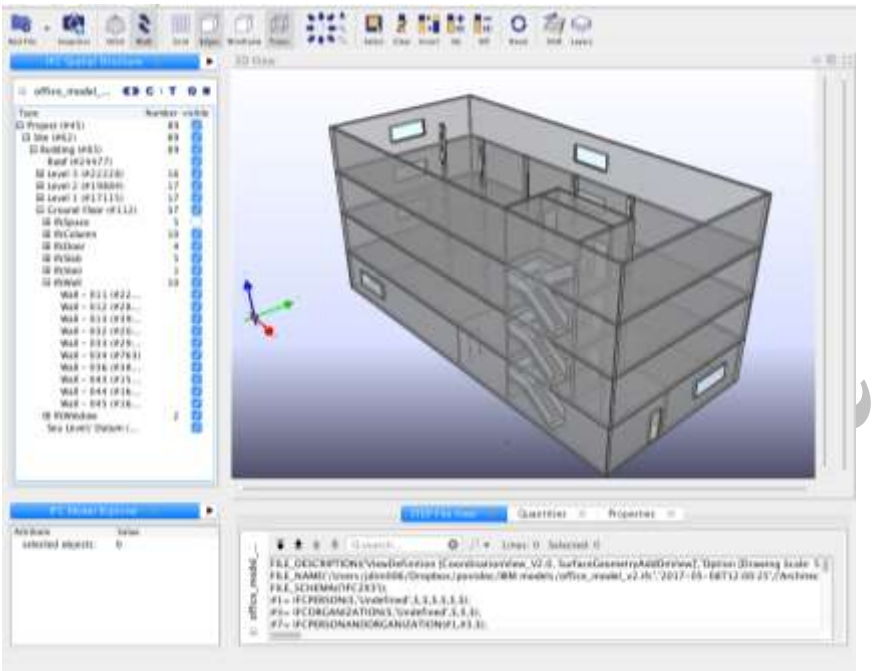

Fig. 5. The IFC model of the sample building in the IfcViewer application

Each level of the building accommodates open plan offices. **Fig. 6** is the ground level layout plan showing two offices separated by an internal wall with an inclined middle section (i.e. "Wall – 044").

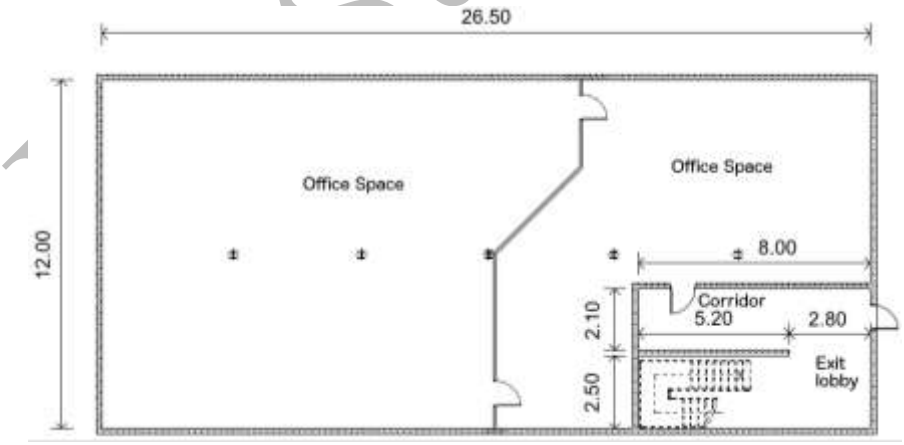

Fig. 6. Ground Level Layout Plan

The upper levels (levels 2 to 4) of the building has a typical floor layout, which is shown in **Fig. 7**. An internal stairwell provides access to all levels through a corridor

on each level. The stairwell also serves as a single means of escape for occupants during an emergency.

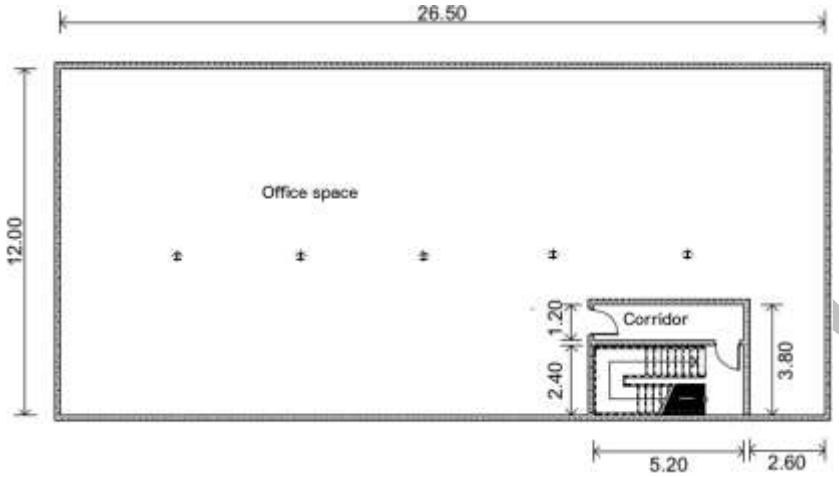

Fig. 7. Typical Upper Levels Layout Plan

5.1 The BIMRL Approach

The IFC model of the sample building is imported into BIMRL and processed by extracting every elements and properties from the model and storing them into the database. Additionally, five geometric representations of each object are generated and stored in the database. **Fig. 8** shows a geometric representation of the entire building model in BIMRL.

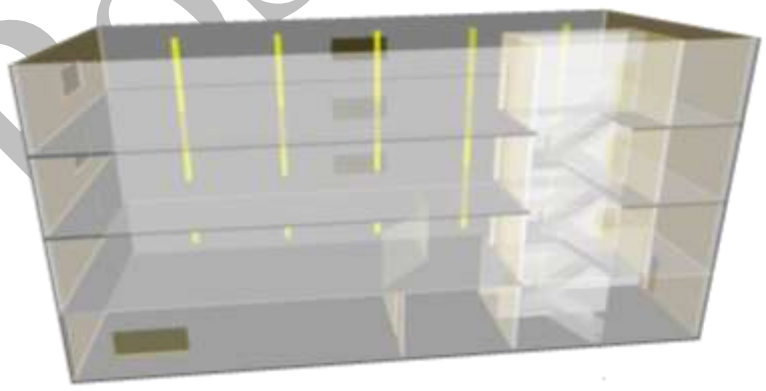

Fig. 8. A geometric representation of the building model in BIMRL

The execution of the query produced a large JSON output, which is due to the large number of voxels generated for the slabs that have a cut-out profile around the stair-well and also due to the representation of the stairs (**Fig. 9**).

Fig. 9. The Octree Cells (level 8) representation of the Stairs

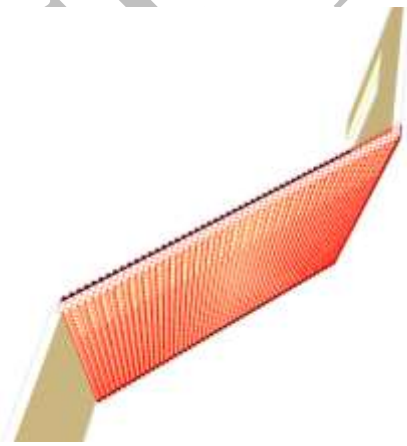

Fig. 10. AABB and Octree Cells representations of different sections of a wall

Below is an excerpt of the query output for the 'Wall – 011' object that is orthogonal with respect to the axes and has an opening, so the AABB geometry representation is appropriate for the object.

```
{
   "ElementId":"19ALpexd7GHBgVjZMl6DWk",
   "ElementType":"IFCWALLSTANDARDCASE",
   "Name":"Wall - 011",
   "Description":"",
   "Materials":[
      {
         "LayerNo":1,
         "Name":"Concrete - Structural",
         "Thickness":0.2
      }
   ],
   "IsOrthogonal":true,
   "IsExternal":false,
   "CellBbox":null,
   "SingleBbox":{
      "Bbox":{
         "LLB":{
            "x":19.2672109375,
            "y":12.5232998046875,
            "z":0
         },
         "URT":{
            "x":27.467150390625,
            "y":12.7232998046875,
            "z":3.2
         }
      },
      "Openings":[
         {
            "SillHeight":0,
            "Bbox":{
               "LLB":{
                  "x":20.54610072752,
                  "y":12.1232997142,
                  "z":0
               },
               "URT":{
                  "x":21.44610072752,
                  "y":12.7232997142,
                  "z":2.1
               }
            }
         }
      ]
```

```
    }
}
```


Another example is ‘Wall – 044’, which is the inclined middle section of the internal timber-framed wall on the ground level, which is subject to voxelization (**Fig. 10**). Below is an excerpt of the query result showing 2 out of 29 voxels generated.


```
{
   "ElementId":"2dZqvRyzIpHuQM7C0wKNZH",
   "ElementType":"IFCWALL",
   "Name":"Wall - 044",
   "Description":"",
   "Materials":[
      {
         "LayerNo":1,
         "Name":"Timber - Wall",
         "Thickness":0.09
      }
   ],
   "IsOrthogonal":false,
   "CellBbox":[
      {
         "LLB":{
            "x":14.464227971062398,
            "y":13.697208381771718,
            "z":-0.0141062499999989
         },
         "URT":{
            "x":14.573928952424797,
            "y":13.757635718506444,
            "z":3.1010312500000000167
         }
      },
      {
         "LLB":{
            "x":14.9073928952424797,
            "y":14.133208381771718,
            "z":-0.0140624999999989
         },
         "URT":{
            "x":15.008629933787197,
            "y":14.234635718506444,
            "z":3.101703125000000167
         }
      }
```

5.2 The Blender Approach

Fig. 11 shows the IFC model of the sample building imported into Blender using the IfcBlender add-on tool.

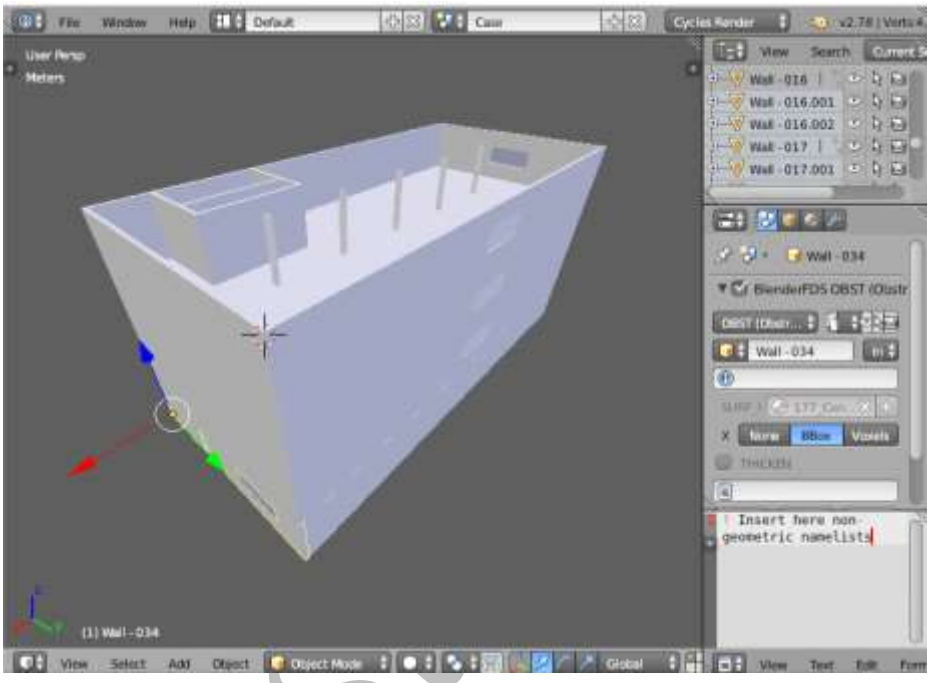

Fig. 11. The building model imported to Blender

The next step is to create the FDS input data by selecting each object in the model individually (**Fig. 12**) using the BlenderFDS user-interface and specify if the object is to be represented as AABB or voxels. Additionally, the material composition (SURF ID namelist group) can be selected for the object, if it has been defined.

Objects can be selected graphically or from the list of objects. Once an object has been classified, its classification and other properties can be copied to other objects.

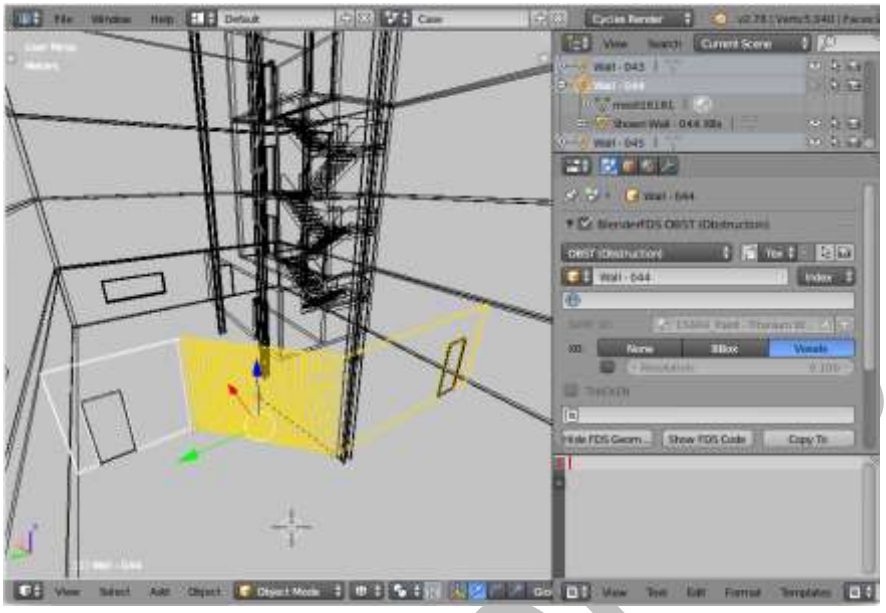

Fig. 12. The non-orthogonal wall being selected in Blender and classified as Voxels

5.3 Generated FDS Input Data

The non-geometrical information such as the simulation parameters are specific to FDS that must either entered by the user, either via a set of default values (using the BIMRL approach) or manually through a user interface (in the case of Blender).

The material composition of walls and its thickness are generally specified in BIM, so the information can be extracted and used in the material property specification in the FDS input file, as shown below. FDS can represent a composite material by specifying the mass fraction of each component, i.e. MATL_MASS_FRACTION.

```
&MATL ID='Concrete - Structural',
      FYI='',
      SPECIFIC_HEAT=1.04,
      CONDUCTIVITY=1.8,
      DENSITY=2280.0/

&SURF ID='Wall - 020',
      COLOR='INVISIBLE',
      BACKING='INSULATED',
      MATL_ID(1,1)='Concrete',
      MATL_MASS_FRACTION(1,1)=1.0,
      THICKNESS(1)=0.20/
```

The thermo-physical properties of the material such as its specific heat, conductivity and density are all assumed initial default values. As a good practice, it is expected that any simulation input data would be checked for validity and appropriateness by the user before the simulation is executed. As part of the validity check, the properties of objects may be changed to more appropriate values depending on the application or scenario.

Below is an excerpt of the generated obstruction specification for 2 different walls in FDS. The first wall, 'Wall – 044', is represented by 29 voxels. The second wall, 'Wall – 020' is represented as a simple AABB.

```
// Wall - 044 represented by 29 voxels
&OBST ID='Wall - 044_0'
      XB=14.464,14.573,13.697,13.757,-0.001,3.101 /
&OBST ID='Wall - 044_1'
      XB=14.907,15.009,14.133,14.235,-0.001,3.101 /


// Wall - 020 external wall
&OBST ID='Wall - 020'
      XB=0.767,27.667,7.523,7.723,3.200,6.400,
SURF_ID='Wall - 020' /
```

Both approaches generated almost identical FDS input files. In general, the BIMRL approach produces a lot more voxels than those by BlenderFDS. This is partly due to the high level of Octree Cells selected for those objects in BIMRL. The generated FDS input file is then executed with a zero simulation duration to produce a set of geometry data that can be visualised using SmokeView (**Fig. 13** and **Fig. 14**).

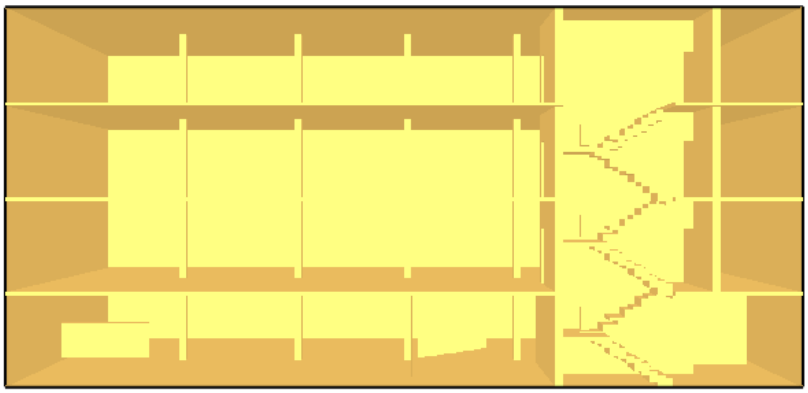

Fig. 13. The sample building model as represented in FDS, viewed in SmokeView

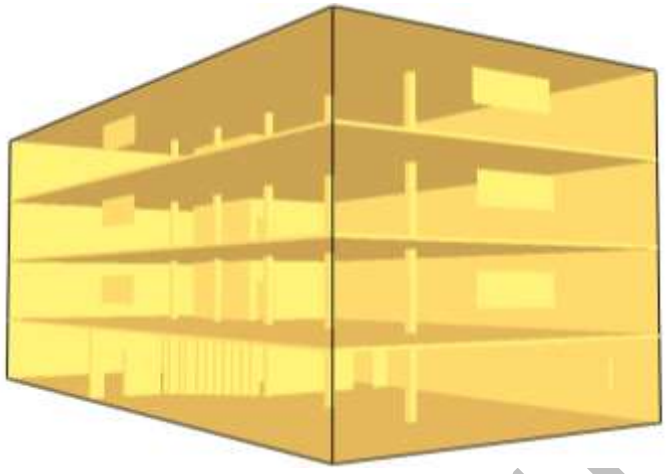

Fig. 14. Another view of the sample building as represented in FDS, viewed in SmokeView

6 Discussions and Conclusion

Two comparative approaches to share IFC geometrical data with the FDS simulation tool has been described in this paper. The BIMRL approach offers a customised query given a data requirements schema, which may include non-geometrical IFC data such as single value properties of certain objects. This is superior to the Blender approach where only geometrical data can be imported. Furthermore, the Blender approach relies on the IfcBlender software add-on with a predefined mapping definition hard-coded into the tool that cannot be modified easily.

Although the BIMRL approach allows extracting some of the semantical data pertinent to fire dynamics simulation such as thermo-physical properties of materials from an IFC model, the actual data may not be present in the architectural model unless it has been enriched with such information by a fire design engineer. Another set of information essential to enclosure fire dynamics simulations is the characteristics of the fire such as its location, type of ignition source, and the potential energy release based on the configuration of combustible materials in the enclosure. Fire design specific parameters are unlikely, if not inappropriate, to be incorporated into the IFC model during the design phase, but would be provided directly into the simulation input dataset as part of a good fire engineering design practice.

The BIMRL approach has an added potential to be integrated with, for example, an automated compliance audit framework that can interface with FDS by generating an input data for it and then using the simulation output (in a comma-delimited format) in a compliance audit process in conjunction with specified normative criteria.

There are two other approaches similar to the Blender's workflow that have not been described in this paper. They involve using commercial tools such as Pyrosim and CYPECAD MED and intermediate data exchanges in proprietary formats, which lacks

transparency. These alternative approaches provide a good user-interface for FDS input data modelling with the advantage of being able to share BIM geometry data.

Future work may include an investigation for an optimised method of representing voxels in BIMRL. Another potential future work is a case study of auditing certain quantities in the FDS simulation output for compliance with specified criteria.

References

1. ISO 16739: ISO 16739:2013 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries. International Organization for Standardization, Geneva, Switzerland (2013)
2. Beetz, J., van Berlo, L.: bimserver.org - An Open Source IFC Model Server. In: Proceedings of CIB W78 International Conference. pp. 1–8. , Cairo, Egypt (2010)
3. Schreyer, A.C.: Architectural Design with SketchUp: 3D Modeling, Extensions, BIM, Rendering, Making, and Scripting. John Wiley & Sons, Inc. (2016)
4. Krijnen, T.: IfcOpenShell, http://ifcopenshell.org/. Last accessed 19 March 2018.
5. Brito, A.: Blender 3D: Architecture, Buildings, and Scenery. Packt Publishing Limited (2008)
6. Dimyadi, J., Eastman, C., Amor, R., Solihin, W., Eastman, C., Amor, R.: Integrating the BIM Rule Language into Compliant Design Audit Processes. In: Proceedings of the 33th CIB W78 International Conference 2016. pp. 1–10. Queensland Institute of Technology, Brisbane, Australia (2016)
7. Baum, H.R.: Simulating Enclosure Fire Dynamics. In: US National Research Council Workshop to Foster Improved Fire Safety. pp. 107–116 (2003)
8. Baker, G., Wade, C., Spearpoint, M., Fleischmann, C.: Developing Probabilistic Design Fires for Performance-based Fire Safety Engineering. Procedia Eng. 62, 639–647 (2013). doi:10.1016/j.proeng.2013.08.109
9. McGrattan, K., McDermott, R., Weinschenk, C., Overholt, K., Hostikka, S., Floyd, J.: Fire Dynamics Simulator User's Guide. National Institute of Standards and Technology (2015)
10. Forney, G.P.: Smokeview (Version 6) A Tool for Visualizing Fire Dynamics Simulation Data Volume II: Technical Reference Guide. (2013)
11. Spearpoint, M.J., Dimyadi, J.: Sharing Fire Engineering Simulation Data Using the IFC Building Information Model. In: International Congress on Modelling and Simulation - MODSIM07. , Christchurch, New Zealand (2007)
12. Dimyadi, J., Spearpoint, M., Amor, R.: Sharing building information using the IFC data model for FDS fire simulation. In: Karlsson, B. (ed.) Fire Safety Science - Proceedings of 9th International Symposium. pp. 1329–1340. IAFSS, Karlsruhe, Germany (2008)
13. Solihin, W., Eastman, C., Lee, Y.C., Yang, D.H.: A simplified relational database schema for transformation of BIM data into a query-efficient and spatially enabled database. Autom. Constr. 84, (2017). doi:10.1016/j.autcon.2017.10.002
14. Solihin, W., Eastman, C., Lee, Y.C.: Multiple representation approach to achieve high-performance spatial queries of 3D BIM data using a relational database. Autom. Constr. 81, 369–388 (2017). doi:10.1016/j.autcon.2017.03.014
15. Peng, D., Cao, L., Xu, W.: Using JSON for data exchanging in web service applications. J. Comput. Inf. Syst. 7, 5883–5890 (2011)
16. Gissi, E., Bartola, R., Santis, N. De, Valpreda, F., Faletti, G., Overholt, K., Dimyadi, J., Orvieto, R.: Blueprint for Blender + FDS. (2009)
17. Thunderhead Engineering: PyroSim: A Model Construction Tool for Fire Dynamics Simulator (FDS). (2010)

18. Dimyadi, J. 3dsolid2fds, https://sites.google.com/site/jdimyadi/3dsolid2fds. Last accessed 19 March 2018.
19. Wolfris: STEP2FDS, https://github.com/wolfris/step2fds. Last accessed 19 March 2018.
20. MBIE: C/VM2 Verification Method: Framework for Fire Safety Design For New Zealand Building Code Clauses C1-C6 Protection from Fire. The Ministry of Business, Innovation and Employment, Wellington, New Zealand (2014)